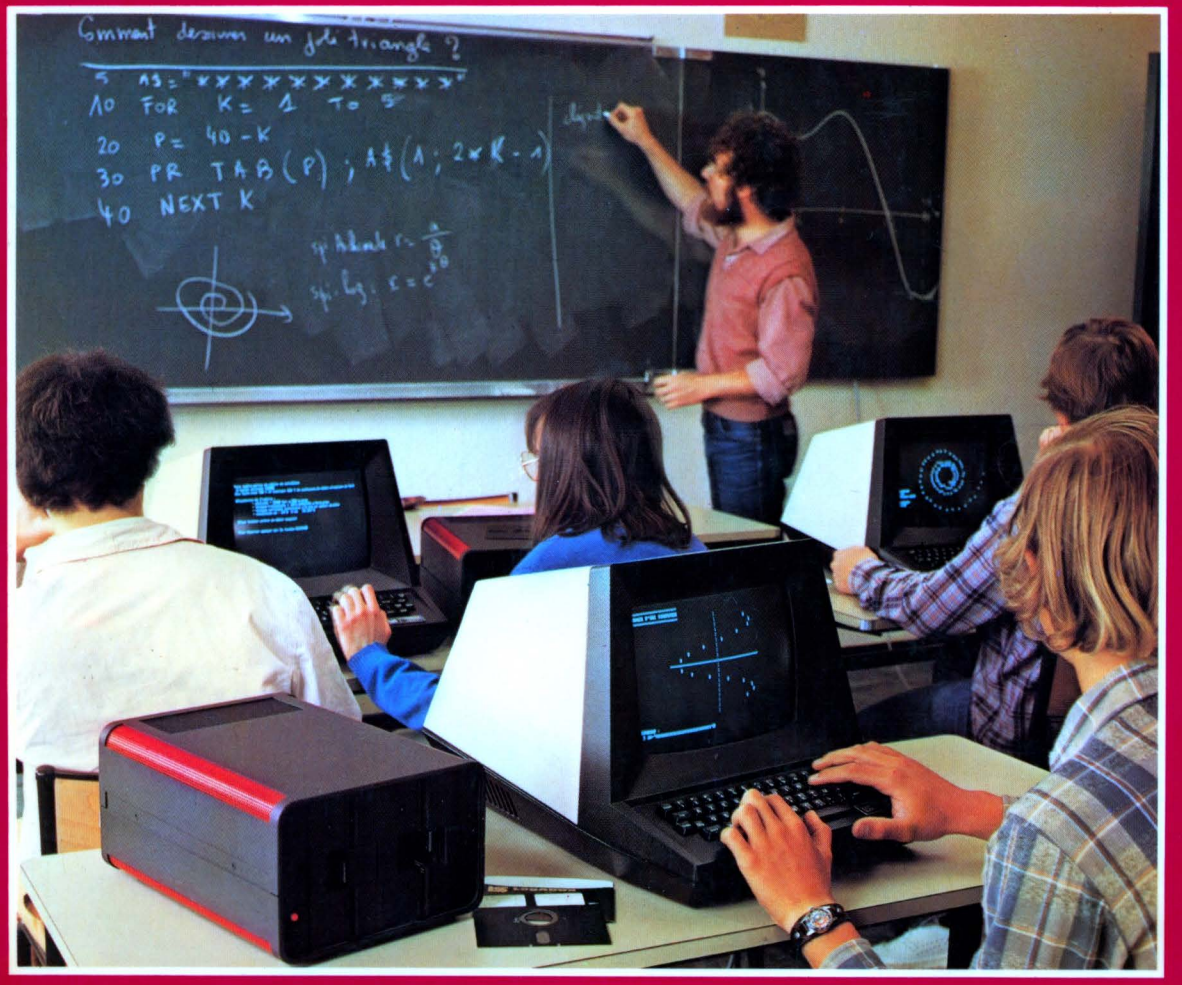
H. LILEN - J.P. NIÈRES - A. POLY

comprendre la MICRO INFORMATIQUE

PRATIQUE DES MICRO-ORDINATEURS



CLASSIQUES - HACHETTE

comprendre la MICRO INFORMATIQUE

Ouvrage réalisé avec la collaboration de la SECF - Editions Radio.

Autres ouvrages de Henri Lilen aux Editions Radio :

- Principes et applications des circuits intégrés linéaires.
- Circuits intégrés numériques.
- Thyristors et triacs.
- Circuits intégrés JFET, MOS, et CMOS (Principes et applications).
- Circuits hybrides à couches minces et à couches épaisses.
- Guide mondial des microprocesseurs.
- Du microprocesseur au micro-ordinateur.
- Mémoires intégrées.
- L'ordinateur personnel.

En collaboration avec R. Arouette :

- Théorie et pratique des microprocesseurs.

comprendre la MICRO INFORMATIQUE

PRATIQUE DES MICRO-ORDINATEURS

Henri LILEN
Directeur-Rédacteur en chef de la revue « Electronique Industrielle ».

Jean-Pierre NIÈRES
Directeur à la société Logabax.

André POLY
Directeur du service informatique à l'Ecole Normale Supérieure de Saint-Cloud.

CLASSIQUES - HACHETTE

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal.
© HACHETTE 1980, 79, boulevard Saint-Germain, F 75006 Paris.
Tous droits de traduction, de reproduction et d'adaptation réservés pour tous pays.

ISBN 2 01 007832 2

LISTE DES CHAPITRES

Note de l'Editeur 7

Chapitre I Le phénomène informatique 9

Chapitre II Informatique et ordinateur 17

Chapitre III Les systèmes de décodage 27

Chapitre IV Opérateurs logiques 51

Chapitre V Description d'un micro-ordinateur 59

Chapitre VI Les périphériques 81

Chapitre VII Analyse d'un système : le LX 500 111

Chapitre VIII Dialogue et programmation 129

Chapitre IX Le système d'exploitation 155

Chapitre X Les programmes et leur exécution 165

Chapitre XI Programmons en basic 179

Chapitre XII Quelques programmes pratiques en Basic 221

Chapitre XIII Du Basic au LSE 247

Chapitre XIV Quelques programmes pratiques en LSE 271

Chapitre XV Maintenance et fiabilité 287

Annexes

1. Le Basic complet 303

2. Commandes de l'interpréteur Basic, de l'éditeur et du BDOS 304

3. Vocabulaire 305

Table des matières détaillée 313

Note de l'Editeur

35 ans après la construction du premier ordinateur, l'informatique déjà présente dans quelques lycées expérimentaux, ainsi que dans bon nombre de lycées techniques et de lycées d'enseignement professionnel, entre en force dans les établissements d'enseignement secondaire : d'ici 1984, 10 000 micro-ordinateurs auront été installés dans les lycées français.

Il ne s'agit pour l'instant, ni d'une nouvelle matière à enseigner, ni d'une nouvelle manière d'enseigner, — mais de la prise en compte dans l'éducation d'un phénomène majeur qui est en train de bouleverser notre société.

L'emploi de l'ordinateur, qui, pendant longtemps a été réservé à la gestion des organismes importants et à la mise en œuvre de grands programmes scientifiques et techniques, tend aujourd'hui à concerner tous les aspects de notre vie. Chacun sait que, si le fait informatique est d'abord un fait technique caractérisé par l'apparition de machines, les ordinateurs sont des machines d'un type nouveau : ils disposent d'une mémoire considérable ; ils peuvent traiter une masse énorme d'informations dans des temps extrêmement brefs. L'enjeu réel est celui d'un nouveau mode d'utilisation des connaissances, et d'un rapport nouveau au savoir et au pouvoir. C'est pourquoi, il apparaît nécessaire aujourd'hui, « d'inscrire la *dimension informatique* dans la phase d'éducation initiale de l'homme, c'est-à-dire dans la phase où il acquiert les modes de raisonnement, les connaissances, et le savoir faire que lui procure l'enseignement ; dans la phase où il intègre naturellement ces divers apports en les plaçant dans le contexte historique, artistique, mais également économique et technologique de son existence ; autrement dit encore, dans la phase où il acquiert naturellement les composantes d'une authentique culture moderne »*.

La manière dont l'informatique peut s'intégrer dans l'enseignement reste l'objet d'un débat. Sans doute l'appareil individuel permet-il de travailler de manière autonome chez soi, ou dans le centre de documentaion d'un lycée ; demain les supports informatiques viendront compléter le livre, au même titre que les vidéocassettes et vidéodisques ; un jour peut-être ordinateur et terminaux individuels se substitueront au tableau noir et aux cahiers. Rien n'assure cependant qu'une telle évolution conduise à un progrès ; les risques sont à la mesure des potentialités.

---

* Jean-René Saurel - Enseignement et informatique, Actes du Colloque international informatique et société, édités par la Documentation française.

Dès aujourd'hui par contre, l'ordinateur peut être l'occasion d'un dialogue renouvelé entre professeurs et élèves, le moyen de manipuler les notions abstraites des mathématiques ou de simuler des expériences de physique, le répétiteur infiniment patient d'un contrôle élémentaire des connaissances. A l'élève lui-même, il permet de faire le point et de rechercher dans une banque de données, une aide qui peut alors être personnalisée. Dans cette perspective, où l'éducation est affaire de comportement individuel et relationnel autant que de connaissances, l'ordinateur a sa place dans une stratégie nouvelle de formation initiale, même s'il ne peut en aucun cas se substituer au professeur.

Ce livre se veut un ouvrage de référence et un ouvrage d'initiation : les notions qui sont développées ici, permettent au lecteur, en fonction de ses préoccupations, soit d'acquérir une culture de base sur l'informatique avec la connaissance des principes et des termes techniques, soit de disposer des éléments d'un savoir-faire suffisant pour l'utilisation courante d'un micro-ordinateur dans l'enseignement.

Enfin, l'édition par les *Classiques Hachette* d'un ouvrage sur l'informatique, constitue un premier pas vers de nouveaux types de publications dans lesquels les supports informatiques auront leur place aux côtés des livres traditionnels.

Chapitre I

LE PHÉNOMÈNE INFORMATIQUE

Faire de l'histoire dans un ouvrage technique, voilà qui peut paraître surprenant. Et pourtant ! Née il y a seulement quelques dizaines d'années, l'informatique ou « ensemble des techniques de traitement de l'information » a, sous la pression technologique, connu un certain nombre de mutations qui toutes ont marqué leur époque, et qu'il est nécessaire de rappeler rapidement pour justifier ce terme « phénomène » que l'on applique aujourd'hui à la micro-informatique.

1. LES ANNÉES 1950 : NAISSANCE DE L'ORDINATEUR

Au début des années 50, de nouvelles techniques mises en œuvre pendant la guerre, à des fins militaires, sont progressivement appliquées au domaine civil. Ainsi en est-il pour la photographie aérienne, les transports aéronautiques, la sécurité en vol avec le radar, le calcul scientifique avec l'ordinateur.

L'ordinateur est véritablement né aux Etats-Unis en 1946. L'ENIAC — *Electronical Numerical Integrator and Calculator* — première calculatrice électronique, s'appuie sur la technologie des tubes. Il en utilise près de 20.000. Il occupe une surface de plusieurs dizaines de mètres carrés, pèse plusieurs tonnes et consomme une puissance électrique importante, de l'ordre de 50 kilowatts. Principalement destiné à des applications scientifiques, l'ENIAC est l'objet des multiples attentions du nombreux personnel qui officie autour de lui. De programmation malaisée, requérant ainsi un personnel spécialisé, ce type de calculatrice est peu fiable. La durée de vie des tubes, ou lampes, n'est que de l'ordre de quelques heures ou dizaines d'heures. L'ENIAC se présente ainsi comme un outil au potentiel extraordinaire mais sa fragilité est extrême.

Cette époque des calculatrices à tubes, (« à lampes »), à base de circuits mettant en œuvre les techniques et les moyens classiques de l'électricité — condensateurs, bobines, résistances — constitue ce que l'on appelle la *première génération des ordinateurs.* Celle-ci apparaît en Europe vers les années 1955. Les premiers utilisateurs sont principalement des organismes importants, à des fins scientifiques essentiellement.

Au plan technique, le temps de réponse des circuits est de l'ordre de quelques microsecondes (1 μs = 1 millionième de seconde). La vitesse de calcul, exprimée en nombre d'opérations élémentaires, est de l'ordre de quelques centaines à quelques milliers par seconde. La mémoire est constituée par un tambour magnétique.

2. L'ARRIVÉE DES SEMI-CONDUCTEURS : DIODES ET TRANSISTORS

La fin des années 50 et le début des années 60 voient l'arrivée sur le marché des transistors qui supplantent rapidement les tubes.

Le temps de réponse des circuits se voit grandement amélioré — quelques centaines de nanosecondes (1 ns = 1 milliardième de seconde) — de même, bien entendu, que la vitesse de calcul : plusieurs centaines de milliers d'opérations élémentaires par secondes.

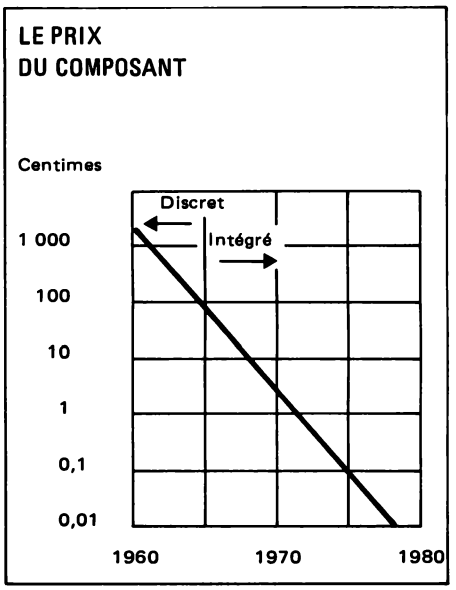
Parallèlement aux progrès accomplis en matière de vitesse de calcul, la fiabilité est multipliée par un facteur 10, les procédures d'entretien s'en trouvant d'autant allégées. La génération des ordinateurs à transistors est appelée « *deuxième génération* ».

3.LA TROISIÈME GÉNÉRATION

Datant du milieu des années 1960, la troisième génération se caractérise par l'appel à diverses techniques dont celle des circuits intégrés (fig. 1.1). Si les mémoires sont encore de type à tores magnétiques, les circuits de calcul, de puissance et les circuits logiques utilisent les progrès fantastiques accomplis en matière de circuits intégrés, circuits réalisés par des procédés divers de dopage (introduction d'impuretés dans des structures cristallines), d'oxydation de surface, de photographie avec masquage, sur un matériau de base constitué par le silicium.

La technologie des circuits hybrides est également utilisée ; des composants discrets (résistances, par exemple) et des semi-conducteurs sont déposés et soudés sur des plaquettes de céramique, le tout étant placé dans une capsule. Dans ce cas, le temps de réponse des circuits élémentaires tombe à quelques dizaines de nanosecondes, parfois à quelques nanosecondes. La vitesse de calcul gagne

Fig. 1-1. — Les circuits intégrés ont été inventés en 1959-1960. On voit ici la courbe qui caractérise l'évolution du degré d'intégration et le nom qui a été attribué aux diverses étapes.



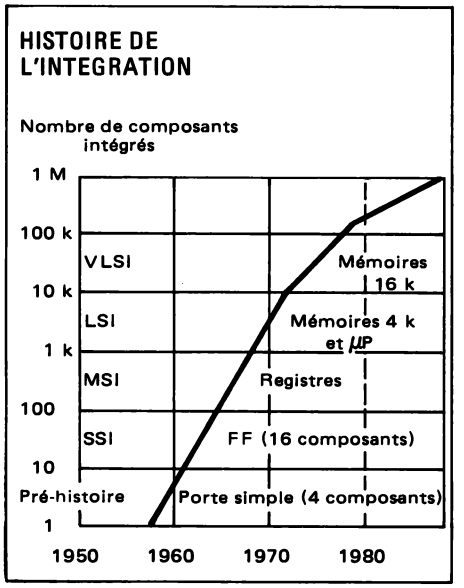


Fig. 1-2. — Lié en tout premier lieu à la surface de silicium utilisé, le prix du transistor intégré n'a cessé de décroître.

encore un ordre de grandeur, le nombre d'opérations élémentaires pouvant être exécutées par seconde passant à plusieurs millions. La fiabilité se trouve également améliorée, de l'ordre d'un facteur 10.

4. APPARITION DES MICRO-ORDINATEURS

L'apparition des micro-ordinateurs est liée aux progrès accomplis en matière de circuits intégrés ; en effet, à l'évolution extraordinaire constatée au plan de *l'intégration* (c'est-à-dire le nombre de composants figurant sur un circuit), s'est ajoutée une grande baisse des prix (fig. 1.2).

Ce phénomène de décroissance des prix a permis aux concepteurs d'ordinateurs de périphériques ou de terminaux, de prendre en compte, quelques années ou quelques mois après leur apparition sur le marché, les différents composants et de les mettre en application dans les produits ou systèmes de leur conception.

Evolution des techniques d'intégration

Nombre de composants par circuit :
- *< 4 : la préhistoire*
- *10 à 100 : étape SSI*
  « single size integration »
- *100 à 1000 : étape MSI*
  « medium size integration »
- *1000 à 10000 : étape LSI*
  « large scale integration »
- *10000 à 100000 : étape VLSI*
  « very large scale integration »
- *de 100000 à un million :*

— étape en cours,
— nouvelles technologies.

Parmi ces composants, citons simplement les *microprocesseurs,* les *mémoires,* les *circuits d'interfaces* — toutes notions que nous retrouverons dans la suite de cet ouvrage.

Le micro-ordinateur construit autour d'un microprocesseur et de divers circuits mémoires, interfaces, etc... apparaîtra au début des années 1970. D'un volume réduit mais d'une puissance — en termes de fonctions réalisées — comparable à celle des ordinateurs de première, deuxième, voire même de troisième génération, son prix relativement bas le mettra à la portée d'utilisateurs de plus en plus nombreux. C'est en cela que réside le phénomène micro-ordinateur : la micro-informatique est une informatique nouvelle, à la portée de tous. Plus facilement accessible en termes de prix comme en termes d'utilisation, elle s'adresse à une clientèle de moins en moins informaticienne, de plus en plus grand public, justifiant ainsi son appellation d'aujourd'hui « informatique personnelle, ou individuelle ».

5. QU'EST-CE QUE LA MICRO-INFORMATIQUE ?

Réfléchissons un instant sur ce terme « Micro-informatique ». Il s'agit tout d'abord *d'informatique,* c'est-à-dire de traitement de l'information. Le micro-ordinateur a ainsi les mêmes fonctions qu'un ordinateur, lequel fait correspondre :

— à des *données introduites,* et à *une description de traitement* à réaliser, des *résultats.*

Quant aux terme *micro,* il caractérise plus le prix et l'encombrement que des limitations fonctionnelles. Un micro-ordinateur est ainsi un ordinateur à part entière.

A l'origine de la micro-informatique se trouve la révolution technologique de la micro-électronique (circuits intégrés, microprocesseurs, mémoires à semi-conducteurs, etc...). La micro-informatique n'est ainsi vieille que de quelques années ; loin de se substituer aux grands ordinateurs, elle a su tout d'abord trouver sa place à ses côtés en s'adressant à des couches de clientèles jusque-là fort peu touchées par l'informatique traditionnelle — principalement en raison de son prix.

Aujourd'hui, l'une et l'autre sont complémentaires, parfois même concurrentes.

6. L'INFORMATIQUE PERSONNELLE

L'informatique personnelle est une informatique autonome, légère, utilisable directement par un opérateur sans autre spécialisation que celle qui consiste à apprendre les principes de base de l'informatique.

Elle est apparue vers 1974 - 1975 aux Etats-Unis sous forme de pièces détachées — cartes micros, cartes mémoires, circuits divers d'interface — celles-ci étaient assemblées par des amateurs pour constituer des systèmes d'automatisation de surveillance, ou de jeux, ou par des professionnels de l'électronique et de l'informatique pour concevoir des systèmes autonomes plus complexes.

De la pièce détachée, on passera rapidement au système complet et, dès les années 1976-1977, des micro-ordinateurs montés, disposant d'un ensemble de programmes, de base et d'application, seront proposés à une clientèle de plus en plus diversifiée. Trois raisons expliquent cette rapide diffusion de l'outil informatique aux plans collectif et individuel :

• *Les prix :* ceux-ci décroissent régulièrement pour atteindre des seuils financiers compatibles avec les ressources du public. Ainsi, une configuration coûte, en 1980, de 25 à 50 000 francs selon le type de programmes d'applications.

• *Les possibilités :* développement des applications dans les domaines les plus divers : calcul, gestion, jeux, automatisation, enseignement assisté, contrôle des connaissances, applications domestiques...

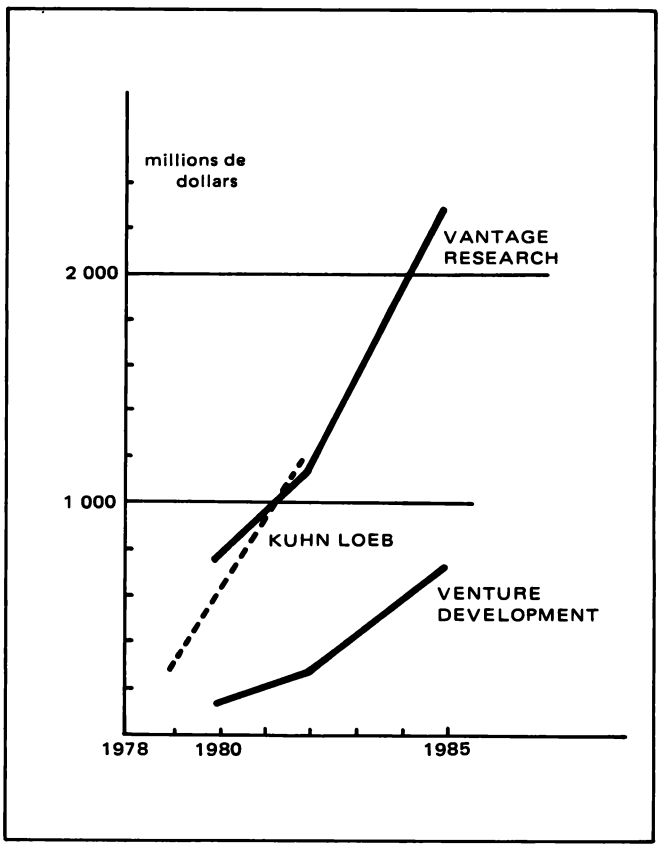


Fig. 1-3. — Quelques évaluations de l'évolution du marché américain des ordinateurs personnels. (Source : Annales des Mines, mars 1979.)

• *Les techniques de commercialisation :* aux réseaux directs de distribution des constructeurs, des assembleurs et des distributeurs d'équipements mécanographiques se sont joints les réseaux classiques de la distribution : boutiques d'abord spécialisées dans l'électronique grand public (télévision, HI-FI), puis boutiques généralistes (électro-ménager, télévision...), grandes surfaces, grands magasins...

Cette multiplication des points de vente n'est d'ailleurs pas encore terminée ; le jour où des micro-ordinateurs seront présentés à la vente par le biais d'une simple correspondance n'est plus tellement éloigné. Le micro-ordinateur rejoindra alors ses cousins technologiques que sont les calculatrices électroniques, les jeux vidéo, programmables ou non, etc...

Premier marché réceptif à ce type d'informatique, les Etats-Unis constituent aujourd'hui la source principale d'innovations et le principal marché mondial (fig. 1.3).

Nombre d'ordinateurs personnels vendus *(Source Venture Developpement Corporation) :*

1975 : 8 000
1976 : 17 000
1977 : 100 000
1978 : 250 000

Les utilisateurs, surtout professionnels de l'informatique ou de techniques-sœurs dans les années 1976, sont aujourd'hui de plus en plus diversifiés — commerçants, gestionnaires, professions libérales, professeurs, étudiants, élèves, amateurs de tous horizons.

Quasiment inexistant en 1976, le marché français a absorbé en 1979 quelque 7 à 8000 systèmes. L'année 1980 devrait voir un doublement, au moins, du parc, les acheteurs se situant aussi bien dans les secteurs public (Administrations, entreprises publiques ou parapubliques, tertiaire...) que privé (petites et moyennes entreprises, petites et moyennes industries, cellules décentralisées de grands organismes ou entreprises, collectivités locales, professions libérales...).

Chapitre II

INFORMATIQUE ET ORDINATEUR

Grands, moyens ou petits, les ordinateurs présentent les mêmes organes essentiels. Les micro-ordinateurs, depuis peu apparus sur le marché mondial, ne font pas exception à la règle. Plus ou moins puissants et rapides, disposant de plus ou moins de possibilités, ils obéissent tous cependant aux mêmes règles :

— l'information qu'ils ont pour charge de traiter est analysée selon les mêmes principes.

— les fonctions qu'ils offrent et les organes qui permettent de les mettre en œuvre sont identiques.

L'informatique, ou « ensemble des techniques et des moyens mis en œuvre pour traiter l'information » *est, ainsi, unique, universelle. Seuls changent les moyens et donc les possibilités. Apprendre à utiliser un micro-ordinateur, c'est accéder aux bases techniques et aux concepts logiques de tout ordinateur : c'est apprendre l'informatique.*

1. L'INFORMATION ET LE TRAITEMENT DE L'INFORMATION

Donner une définition précise de ce qu'est l'information est assez malaisé, tout au moins si l'on veut ne pas se limiter à une définition simple. Aussi, fournirons-nous plutôt quelques exemples qui permettront de saisir, d'approcher le concept d'information.

Exemple 1
Imaginons que nous ayons à effectuer l'opération suivante :

$$X = 2 + 0{,}5 \times Y$$

pour Y = 2. La formule fait apparaître :

— Une partie à droite, décrivant le traitement à effectuer. *Celle-ci comprend, dans notre exemple, des nombres et ou des lettres, soit des constantes et ou des variables, ainsi que des signes représentatifs des opérations à effectuer.*

On retrouve là les notions bien connues d'opérandes et d'opérateurs. Les opérandes et opérateurs nous sont fournis pour que nous soyons à même de faire l'ensemble des opérations nécessaires au calcul de la formule. Ils constituent des informations.

— Une partie gauche, dénommée X, dans laquelle sera stocké le résultat.

Le calcul proprement dit constitue « le traitement » ; son résultat fournit la valeur recherchée, celle de la variable X.

Exemple 2
Je viens de recevoir une lettre d'un ami.Celle-ci se compose d'une enveloppe portant mon nom et mon adresse, et de la lettre proprement dite. Ouvrir l'enveloppe, lire la lettre, c'est accéder à l'information, et en prendre connaissance : j'ai traité de l'information.

Le préposé aux P. et T., pour être en mesure de délivrer cette lettre, a eu, aussi, à prendre connaissance de mon adresse. L'adresse, avec mon nom, le nom de la rue, le numéro dans la rue, d'autres informations peut-être, constituent de l'information. *En la lisant, le préposé l'a interprétée, puis traitée.*

Ces deux exemples traduisent à l'évidence deux choses :

- l'information est omni-présente,
- en permanence, nous traitons de l'information.

Mécanisme de traitement de l'information

Nous nous appuierons sur l'exemple 1 pour introduire les éléments essentiels de ce mécanisme.

La formule à calculer et la valeur de Y ont été transcrites sur une feuille de papier ; le papier est le support de la formule, un support permanent (fig. 2.1).

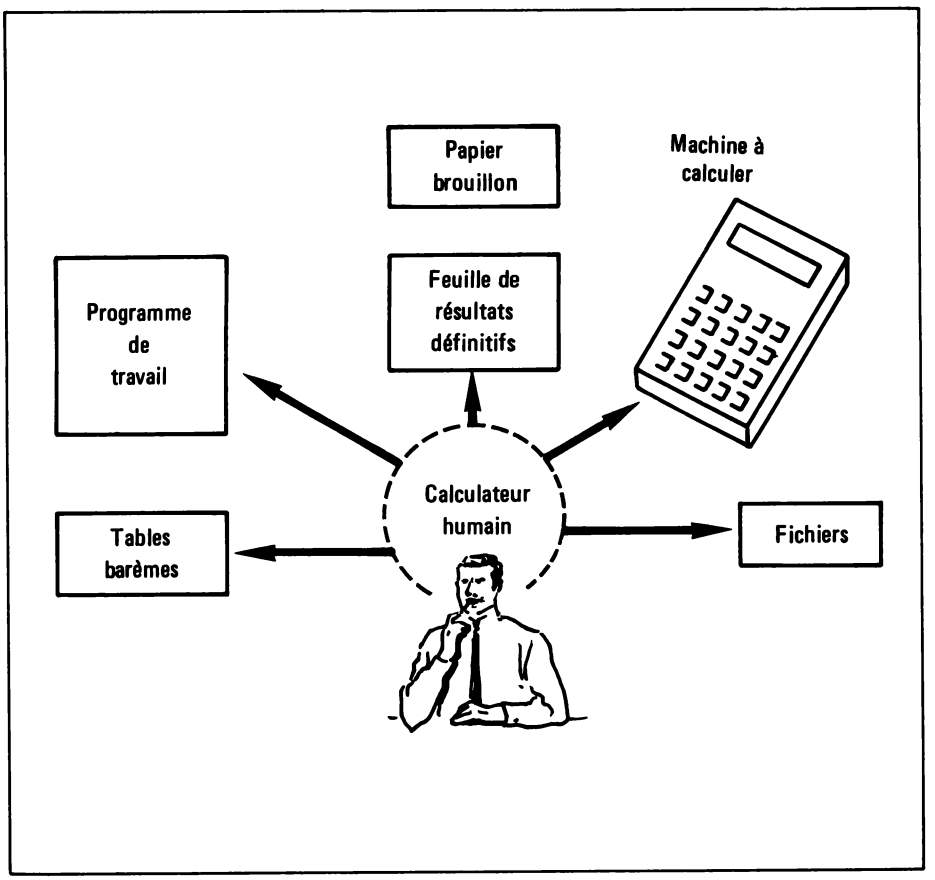


Fig. 2-1. — Les fonctions exécutées dans le déroulement d'un travail sont les fonctions : calcul (la machine à calculer) ; conservation de l'information (programme de travail, feuille de brouillon, tables et barèmes, fichiers, résultats définitifs) ; saisie de données et écriture des résultats ; enchaînement (l'homme).

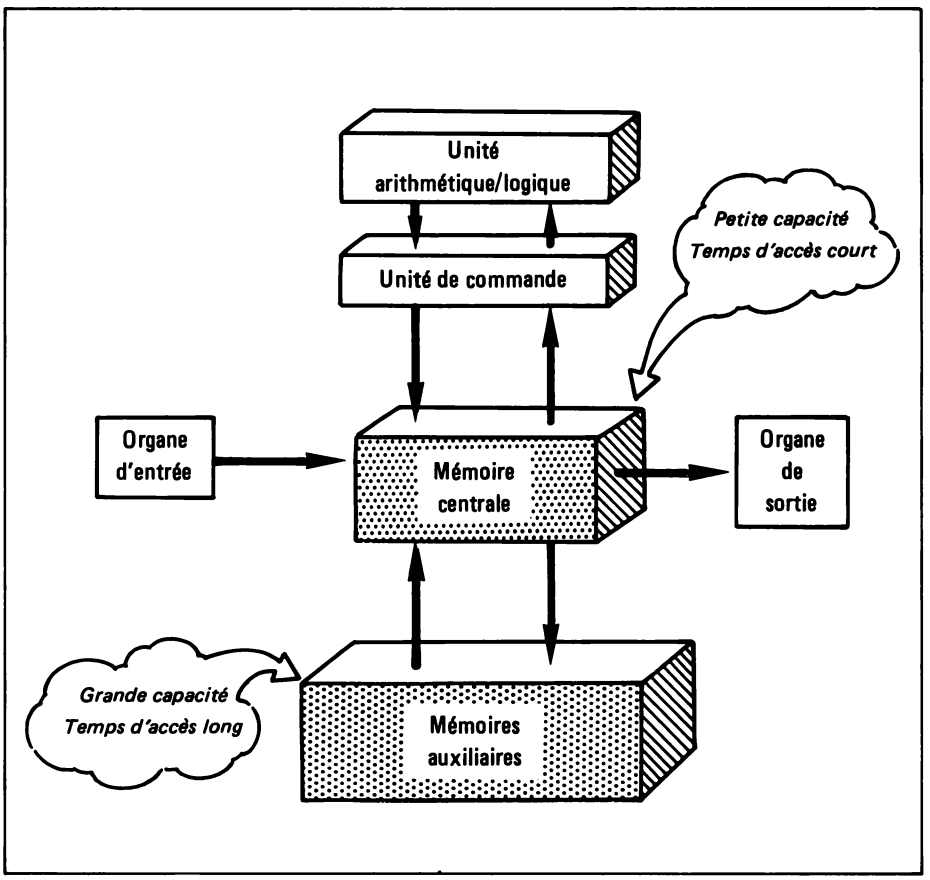


Fig. 2-2. — Structure d'un ordinateur.

Calculer la formule revient à lire ses différents éléments, à les interpréter, à calculer, à écrire enfin le résultat. Le calcul proprement dit a consisté en :

— la lecture de la valeur de Y ;
— sa multiplication par 0,5 (ce qui implique la lecture de 0,5 et celle du signe x ainsi que leurs interprétations) ;
— l'addition de la constante 2 avec le résultat précédent ;
— l'affectation à X (on dira aussi le stockage en X) du résultat obtenu, c'est-à-dire l'écriture.

Trois fonctions essentielles sont ainsi mises en évidence :

- Lecture ;
- Calcul ;
- Ecriture ;

auxquelles il y a lieu d'ajouter la *fonction de mémorisation* mise en œuvre lors de l'écriture, tant de la formule que des résultats intermédiaires et final.

Le micro-ordinateur possède toutes ces fonctions et les met en œuvre dans toute opération de traitement de l'information (fig. 2.2) :

- La formule est stockée dans la *mémoire.*
- La recherche de la formule, son interprétation et son calcul sont réalisés par un organe de logique et de calcul appelé *unité centrale.*
- Lire, écrire, constituent des *entrées-sorties.*

Toutes ces fonctions s'appuient sur des dispositifs spécialisés que nous allons maintenant examiner.

2. ARCHITECTURE D'UN ORDINATEUR

L'unité centrale, la mémoire et les dispositifs d'entrée et de sortie de données constituent les principaux sous-ensembles d'un ordinateur, quel qu'il soit.

2.1 La mémoire

L'ordinateur est un outil. Sous contrôle humain, il travaille selon un schéma pré-déterminé sur des données bien précises. Les exemples 1 et 2 nous ont donné deux illustrations de tels schémas ; ce schéma décrit l'ensemble des opérations à effectuer et sur quels opérandes ces opérations vont porter. La description correspond ainsi en quelque sorte à la notion de programme de travail. Par la suite, nous parlerons de *programme.* Le fait de décrire ces opérations aboutit à la notion de *programmation ;* l'homme qui écrit le programme est un *programmeur.*

Les opérandes constituent des *données ;* on verra ultérieurement dans le chapitre consacré à la programmation que ces données peuvent être de plusieurs types.

Programmes et données, pour être traités, doivent être stockés sur un support. Ce support est appelé *mémoire.* La mémoire répond à trois fonctions :

— enregistrement ;
— conservation ;
— restitution.

Elle présente en outre deux caractéristiques :
— *sa capacité* (ou dimension), c'est-à-dire le nombre de caractères pouvant y être enregistrés et conservés pour une restitution ultérieure ;

— *son temps d'accès,* c'est-à-dire le temps nécessaire pour que, une demande de restitution étant faite, l'information soit localisée et effectivement transmise.

Capacité et temps d'accès

Elles dépendent considérablement du type d'ordinateur.

Pour un micro-ordinateur, une capacité typique est de l'ordre de 16 à 64 K, 1 K correspondant à 1024 caractères. Un caractère est exprimé par une configuration binaire, en général 8 ; cette configuration à 8 bits est appelée octet.

Un micro-ordinateur typique tel que le LX 500 de *Logabax,* par exemple, offre une capacité mémoire qui peut évoluer de 16 à 64 K octets Le temps d'accès, quant à lui, est de l'ordre de quelques microsecondes (millionièmes de seconde).

2.2 L'unité de traitement

Un programme est constitué d'une séquence d'instructions élémentaires ; ainsi, la formule $X = 2 + 0{,}5 \times Y$ est calculée progressivement selon l'enchaînement suivant :

- prendre la valeur de Y (soit 2) ;
- la multiplier par 0,5 (soit 1) ;
- ajouter 2 (soit 3) ;
- mettre le résultat dans X (soit 3).

Quatre instructions élémentaires sont ainsi nécessaires pour calculer notre formule. L'exécution de chacune d'elles a mis en œuvre un ensemble de circuits électroniques spécialisés dans le décodage, le calcul, le transfert, etc... d'information. Cet ensemble constitue *l'unité centrale de traitement* (UC) de l'ordinateur ; c'est le cœur même de l'ordinateur.

*Le décodage de l'instruction a pour but de reconnaître quel type d'instruction est à réaliser.

C'est là l'objet de l'*unité de commande.*

*Une fois reconnue, l'exécution est à réaliser.

Le sous-ensemble mis en œuvre est l'unité arithmétique et logique.

Arithmétique :
L'instruction à exécuter est une opération arithmétique :
— *addition*
— *soustraction*
— *multiplication*
— *division.*

Logique :
L'instruction est une opération à caractère logique :
— *inférieure à...*
— *supérieure à...*
— *etc...*

L'unité de commande et l'unité arithmétique et logique constituent l'unité centrale de traitement.

2.3 Les entrées-sorties

Reprenons l'exemple de notre lettre. Pour que nous ayons pu recevoir notre lettre, le préposé a dû prendre connaissance de l'adresse figurant sur l'enveloppe. Il a lu l'adresse, a interprété les informations la constituant en reconnaissant, par exemple, la rue, le numéro dans la rue.

Cette opération de prise de connaissance s'est manifestée par une *lecture ;* pour savoir quoi faire, l'ordinateur doit aussi *lire les informations.* C'est là l'objet de dispositifs particuliers appelés « dispositifs d'entrée de données » ou de « lecture de données ».

Inversement, lorsque le résultat d'une opération est atteint, il est nécessaire de le restituer au demandeur ; ce résultat pourra être écrit sur une feuille de papier, perforé sur une carte, enregistré sur une bande magnétique.

C'est là une opération *d'écriture* qui met aussi en œuvre un dispositif particulier.

Exemples de dispositifs :
— machine à écrire, imprimante ;
— écran cathodique ;
— lecteur de cartes ;
— clavier ;
— perforateur de cartes.

Le plus souvent, les fonctions lecture et écriture coexistent dans le même dispositif ; ce dispositif est ainsi un dispositif d'entrée et de sortie de données.

Exemples :
- *Lecteur/enregistreur de bandes magnétiques.*
- *Ecran/clavier.*
- *Unité de disques.*
- *Lecteur/perforateur.*
- *Clavier/imprimante.*

Les micro-ordinateurs, tout comme leurs aînés, s'appuient sur de tels dispositifs pour leurs entrées/sorties. Les plus fréquemment utilisés aujourd'hui sont :

- le dérouleur à cassettes magnétiques ;
- le lecteur/enregistreur de disques souples ;
- l'écran/clavier ;
- le clavier/imprimante.

Parmi ces dispositifs, certains comme la bande ou le disque magnétique permettent aussi le stockage de l'information. S'ajoutant à la mémoire principale de l'ordinateur, ils constituent de ce fait des mémoires *auxiliaires,* ou encore *secondaires.*

Nous sommes ainsi en mesure de représenter, sous forme de schéma, l'architecture d'un ordinateur, avec ses principaux composants et ses principales fonctions (fig. 2.3).

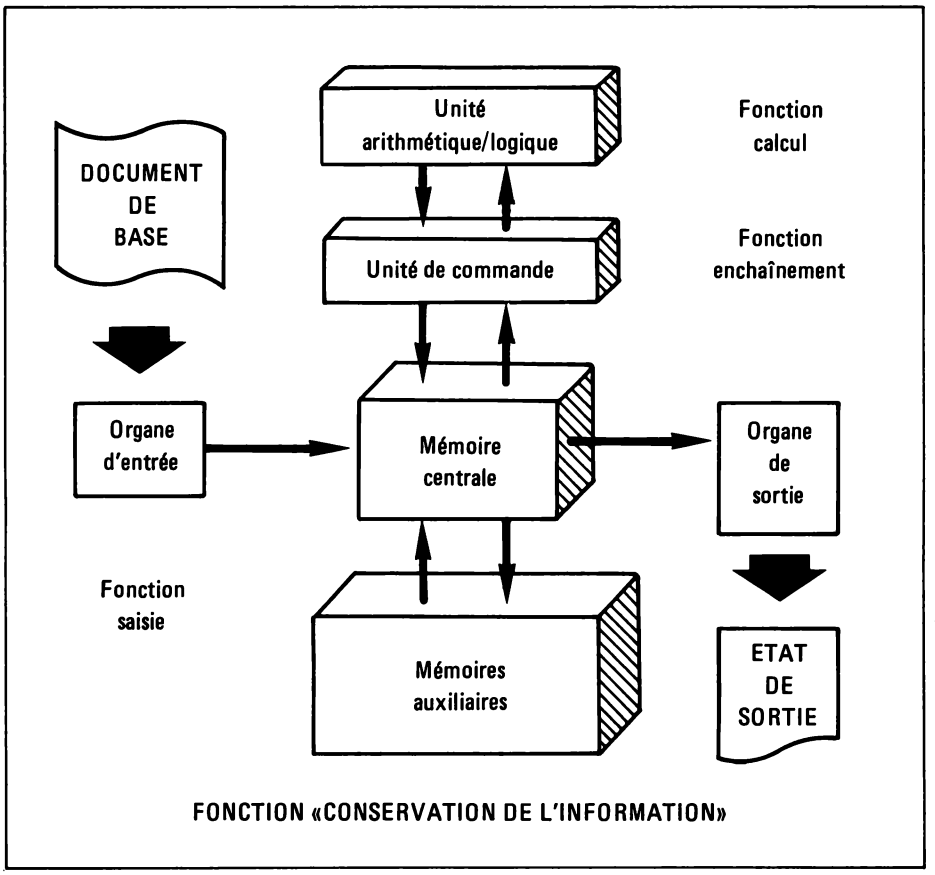


Fig. 2-3. — Les fonctions d'un système informatique.

3. L'ORDINATEUR ET LES ECHELLES DE TEMPS

Il a été fait à plusieurs reprises référence à des ordres de grandeur de temps.

Voici quelques ordres de grandeur pour le traitement d'un caractère par les différents sous-ensembles :

Unité de traitement : 0,1 microseconde
— Mémoire centrale : 1 microseconde
— Dérouleur de bande magnétique : 20 microsecondes
— Lecteur de carte : 2500 microsecondes.

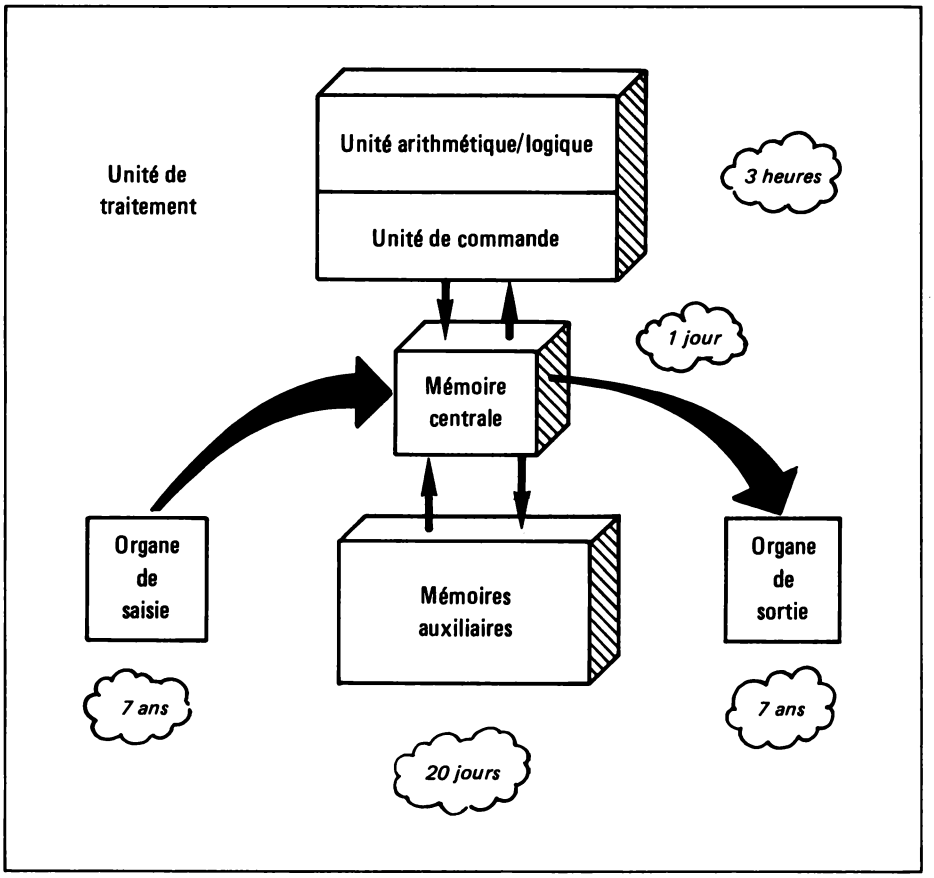


Fig. 2-4. — Si l'on transposait la microseconde (millionnième de seconde) en un jour, voilà ce que donneraient les temps typiques de la machine informatique.

Si l'on convertit ces temps en JOURS, cela donnerait ce que montre la figure 2.4

Le rapport entre
une micro-seconde et une seconde
est le même qu'entre
un jour et 3 000 ans

Chapitre III

LES SYSTÈMES DE CODAGE

1. NOTION D'INFORMATION DIGITALE *

La technique des ordinateurs repose généralement sur l'utilisation d'une représentation *digitale* des informations ; elle s'oppose à une représentation *analogique.*

Analogique, digital, quelles sont les significations de ces deux termes ?

> ***Analogique :***
>
> *L'information est constituée par les variations continues d'une grandeur, entraînant à leur tour des variations continues d'intensité ou de tension électrique. La représentation sera d'autant plus satisfaisante que les variations seront proportionnelles entre elles.*
>
> ***Digital :***
>
> ***L'information est représentée par un système*** *à* n *états d'équilibres, chaque état d'équilibre correspondant à une valeur de l'information.*
>
> *Par définition, l'information digitale est* discontinue.

(*) Une bonne utilisation de la langue française voudrait que le terme numérique soit utilisé. Le terme digital est cependant très usité, aussi le conservons-nous ici.

Les systèmes de traitement de l'information utilisent en pratique un système à 2 états d'équilibre. Une information peut alors revêtir 2 valeurs possibles auxquelles on peut associer une signification.

Les significations associées peuvent être :

— la valeur 0 et la valeur 1 :

• c'est là la base du fonctionnement de l'unité arithmétique

— les termes « VRAI » ou « FAUX » ;
« OUI » ou « NON » ;
« POSSIBLE » ou « IMPOSSIBLE »...

• il s'agit alors de la base de fonctionnement des tests logiques mis en jeu par l'unité logique.

Le système de numération fondé sur l'existence possible de deux états est dit *système binaire ;* il est à la base du fonctionnement des calculateurs digitaux présents aujourd'hui sur le marché.

Un système à deux états est dit binaire. Par convention les deux états sont notés 0 et 1

La vie quotidienne nous donne des exemples courants de systèmes binaires. Ainsi, une lampe électrique peut être éteinte (état 0) ou allumée (état 1) ; une porte peut être ouverte ou fermée, un individu présent ou absent, etc.

2. BASE ET SYSTÈME DE NUMÉRATION

2.1. Le système décimal

Depuis notre prime enfance, nous sommes habitués à un système autre que le système binaire ; il s'agit du système décimal.

Cette décomposition de l'exemple (page suivante) fait apparaître les notions d'*unité,* de *dizaine,* de *centaine.*

Exemple :

L'analyse d'un nombre quelconque, comme 237 par exemple, dans un système décimal (système dit à base 10) et dans lequel nous voulons faire apparaître les différentes puissances de 10, aboutit à la formule

$$237 = 200 + 30 + 7$$
$$= 2 \times 10^2 + 3 \times 10^1 + 7 \times 10^0$$

Unités : 7, dizaines : 3, centaines : 2

Une unité d'un ordre quelconque vaut dix unités de l'ordre immédiatement inférieur.

Exemples

1 - la dizaine vaut dix fois l'unité ;
2 - la centaine vaut dix fois la dizaine ;
3 - le millier vaut dix fois la centaine.

On comprend ainsi aisément pourquoi *10* constitue la *base* du système.

2.2. Système à base B

Dans un tel système et conformément à ce qui vient d'être vu, un système à base B est un système dans lequel les unités des divers ordres sont de B en B fois plus grandes les unes que les autres :

- l'unité du premier ordre est : $1 = B^0$
- l'unité du deuxième ordre est : $B^0 \times B = B^1$
- l'unité du troisième ordre est : $B^1 \times B = B^2$

2.3. Formule de définition d'un nombre

L'exemple étudié au 2.1. nous a fourni la décomposition du nombre 237 selon les puissances de 10 :

$$237 = 2 \times 10^2 + 3 \times 10^1 + 7 \times 10^0$$

Ainsi, un nombre quelconque N inférieur à 100, se décompose selon les puissances de 10, par la formule :

$$N = a \times 10^2 + b \times 10^1 + c \times 10^0$$

et, plus généralement, si la base de notre système est appelée B, un nombre exprimé en base B peut alors se décomposer selon la formule :

$$N = a_{n-1}.B^{n-1} + a_{n-2}.B^{n-2} + \ldots\ldots a_1B^1 + a_0.$$

> ***Remarque :***
>
> $a^0 = a^0.B^0$
>
> *car* $B^0 = 1$ *quel que soit B.*

Cette formule est valable quelle que soit la base de numération du système, *binaire, décimale, octale, hexadécimale...*

> ***Exemples de systèmes de numération :***
>
> - *Base 2 : système binaire ;*
> - *Base 5 : système quinaire ;*
> - *Base 8 : système octal ;*
> - *Base 10 : système décimal ;*
> - *Base 16 : système hexadécimal.*

2.4. Différents systèmes de numération

Chaque système de numération exige un nombre de caractères égal au nombre d'unités contenues dans cette base. Le plus familier d'entre tous, le système décimal, va nous servir à illustrer cette définition.

• *Le système décimal*

La base B vaut donc 10.
L'unité de rang 1 — la dizaine — contient 10 unités.
De même, l'unité de rang 2 — la centaine — contient dix unités de rang 1 ou dizaines.
Ainsi, le système décimal, ou à base 10, utilise 10 caractères différents. Ces caractères sont l'un quelconque des dix chiffres suivants :

0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9

• *Le système binaire*

La base vaut 2.
Les caractères utilisés sont 0 et 1.

• *Le système octal*

La base vaut 8.
Les caractères utilisés sont 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7

• *Le système hexadécimal*

La base vaut 16.

Dans tout système de base supérieure à 10, on notera que, par convention, la série des chiffres vulgaires est complétée par les premières lettres de l'alphabet.

Hexadécimal

Les caractères utilisés dans le système hexadécimal sont ainsi :
1 à 9 pour les 9 premiers
A pour le 10e
B pour le 11e
C pour le 12e
D pour le 13e
E pour le 14e
F pour le 15e
0 pour le 0

Ces divers types de numération — octal et hexadécimal — sont les plus répandus en informatique. Le tableau ci-après donne les valeurs équivalentes des 16 premiers nombres décimaux en binaire, octal et hexadécimal.

Quatre systèmes de numération utiles en informatique

DÉCIMAL	BINAIRE NATUREL	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	1 0000	20	10

2.5. Conversion de base à base

Passage du décimal au binaire

La règle consiste à décomposer le nombre exprimé en décimal en ses successives puissances de 2.

Exemples :

a) $5 = 4 + 1$
$= 2 \times 2 + 1$
$= 2^2 + 2^0$
$= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
$= 1\ 0\ 1$

b) $0 = 0 \times 2^0 = 0$
Le zéro a la même représentation en binaire qu'en décimal.

c) $16 = 2 \times 8$
$= 2 \times 2 \times 2 \times 2$
$= 2^4$
$= 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
$= 1\ 0\ 0\ 0\ 0$

Passage du décimal en hexadécimal

La règle consiste à décomposer le nombre en puissances successives de 16.

Exemples :

1) $16 = 16^1$
$= 1 \times 16^1 + 0 \times 16^0$
$= 10$

2) $12 = 0 \times 16^1 + 12 \times 16^0$
or 12 en base 16 s'écrit C
$12_{10} = C_{16}$

3) $7 = 0 \times 16^1 + 7 \times 16^0$
$= 7$

4) $0 = 0 \times 16^0 = 0$

Règles :

Nous avons vu que le nombre 0 en décimal s'exprimait de la même façon en binaire : il en est de même lors du passage en hexadécimal. On verra aisément que le résultat est identique lors du passage en octal.

En règle générale, le passage du système décimal au système à base *B* laisse les représentations des *B-1* premiers chiffres inchangés.

Exemples :

B = 2	*B-1 = 1*	*chiffres inchangés : 0 et 1*
B = 8	*B-1 = 7*	*" " : 0,1, ... 7*
B = 16	*B-1 = 15*	*" " : 0,1, ... F*

Passage du décimal en binaire

Selon les règles générales exprimées précédemment nous obtenons par exemple :

$$\begin{aligned} 5 &= 4 + 1 \\ &= 2 \times 2 + 1 \\ &= 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 1\ 0\ 1 \end{aligned}$$

Une autre forme de procéder consiste à diviser par les puissances de 2

$$\begin{aligned} 5 : 2 &= 2 \quad + 1 \text{ de reste} \\ 2 : 2 &= 1 \quad + 0 \text{ de reste} \\ 1 : 2 &= 0 \quad + 1 \text{ de reste} \end{aligned}$$

le résultat demandé est : 101

2.6. Les nombres fractionnaires

Un nombre fractionnaire N peut être exprimé dans un système à base B selon la formule suivante :

$$\boxed{N = a_1.B^{-1} + a_2.B^{-2} \ldots . + a_n.B^{-n}}$$

avec $B^{-1} = 1/B^1$
$B^{-2} = 1/B^2$
$B^{-n} = 1/B^n$

Soit un nombre fractionnaire A exprimé en base B et s'écrivant sous la forme 0, $N_1\ N_2\ N_3$

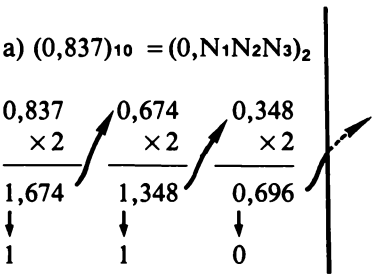
$$A = \frac{N_1}{B^1} + \frac{N_2}{B^2} + \frac{N_3}{B^3}$$

$$\begin{aligned} \text{on a : } A \times B &= N_1 + \frac{N_2}{B^1} + \frac{N^3}{B^2} \\ &= N_1 + A' \\ A' \times B &= N_2 + \frac{N_3}{B^1} \\ &= N_2 + A'' \\ A'' \times B &= N_3 \end{aligned}$$

Le mécanisme de conversion est alors simple :

Exemples :

a) $(0,837)_{10} = (0,N_1N_2N_3)_2$



$(0,837)_{10} = (0,110)_2$ par approximation

b) $(0,101)_2 = \frac{1}{2^1} + \frac{0}{2^2} + \frac{1}{2^3}$

$= \frac{1}{2} + 0 + \frac{1}{8}$

$= 0,5 + 0 + 0,125$

$= (0,625)_{10}$

Tableau des puissances de 2

n	2^n	2^{-n}
0	1	1
1	2	0,5
2	4	0,25
3	8	0,125
4	16	0,062 5
5	32	0,031 25
6	64	0,015 625
7	128	0,007 812 5
8	256	0,003 906 25
9	512	0,001 953 125
10	1024	0,000 976 562 5
11	2048	0,000 488 281 25
12	4096	0,000 244 140 625

3. ARITHMÉTIQUE BINAIRE

Ce système, vu partiellement jusqu'ici, a été introduit au 17e siècle par le mathématicien LEIBNITZ.

3.1. Table d'addition

Elle obéit aux règles suivantes :

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=0$ + retenue soit 10

3.2. Table de multiplication

$0\times0=0$
$0\times1=0$
$1\times0=0$
$1\times1=1$

L'application des règles de l'addition avec décalage successif constitue la règle la plus simple, on peut aussi procéder par additions successives.

Exemples :

```
   1 0 1  (5)          1 1 1  (7)
 + 0 1 1  (3)        + 0 0 1  (1)
 -------             ---------
 1 0 0 0  (8)        1 0 0 0  (8)

   1 0 1  (5)
 × 0 1 1  (3)
 -------
     1 0 1
   1 0 1
 0 0 0
 -----------
 0 1 1 1 1  (15)
```

3.3 Soustraction binaire

On peut utiliser les mêmes règles que pour l'addition, par exemple :

$$\begin{array}{r} 1\ 0\ 1 \\ 0\ 1\ 1 \\ \hline 0\ 1\ 0 \end{array}$$

La méthode par complémentation constitue une autre façon de soustraire :

Complément à 1 (ou complément restreint)

On forme le complément à 1 d'un nombre binaire en changeant

- les 0 de cette configuration en 1 ;
- les 1 de cette configuration en 0.

Le complément à 1 du nombre binaire 101 est 010 ; le complément de 011 est 100.

DÉCIMAL	VALEUR ABSOLUE ET SIGNE	COMPLÉMENT RESTREINT (à 1)	COMPLÉMENT VRAI (à 2)
+ 1	0 0001		
+ 0	0 0000	0 0000	0 0000
— 0	1 0000	1 1111	0 0000
— 1	1 0001	1 1110	1 1111
— 2	1 0010	1 1101	1 1110
— 7	1 0111	1 1000	1 1001
— 8	1 1000	1 0111	1 1000

Complément à 2 (ou complément vrai)

Il est obtenu en ajoutant 1 au complément à 1

Définitions

- Le complément à 1 est appelé complément restreint.
- Le complément à 2 est appelé complément vrai.

Application à la soustraction :

Pour soustraire deux nombres binaires, il suffit d'ajouter le complément à 2 du nombre à soustraire au premier nombre et de ne pas prendre en compte la dernière retenue. En reprenant le même exemple que ci-dessus, on a ainsi :

Complément à 1 de 011 : 100
Complément à 2 de 011 : 100 + 1 = 101

Soustraction de
$(101)_2 - (011)_2 = (101)_2 + (101)_2 =$

$$\begin{array}{r} 101 \\ +\ 101 \\ \hline 1010 \end{array}$$

dernière retenue à éliminer

Le résultat est donc 010.

3.4. Division binaire

Elle est exécutée comme la division décimale, en d'autres termes, elle se décompose en une suite de soustractions et de décalages vers la droite :

```
 1 0 0 1 0 1 0  |  1 0 1
                |
-0 0 0          |  0 1 1 1 0
 -------        |
 1 0 0 1        |
-1 0 1          |
 -------        |
  1 0 0 0       |
 -1 0 1         |
  -------       |
  0 1 1 1       |
 -1 0 1         |
  -------       |
  0 1 0 0       |
 -0 0 0         |
   1 0 0        |
```

4. NUMÉRATION HEXADÉCIMALE

L'écriture binaire est très rapidement fastidieuse et d'un emploi peu commode ; aussi regroupe-t-on les informations binaires ou « bits » par groupes.

En hexadécimal, les chiffres binaires sont ainsi regroupés par 4 :

Binaire	Hexadécimal	Décimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15
1 0000	10	16

Exemples :

a) Passage de l'hexadécimal vers le binaire :

1	*A*	*0*
0001	*1010*	*0000*

5	*B*	*7*
0101	*1011*	*0111*

b) Passage du binaire vers l'hexadécimal :

• *0001 1010 0000 = (0001) (1010) (0000)*
soit, selon la table précédente
1 A 0

• *0101 1011 0111 = (0101) (1011) (0111)*
= 5 B 7

c) Vérification en décimal :

• $1\ A\ 0_{16} = 1\times16^2+(10)\times16^1+0\times16^0$
$=256+160+0$
$=416$

• $0001\ 1010\ 0000 = 1\times2^5+1\times2^7+1\times2^8$
$= 32 + 128 + 256$
$= 416.$

5. CARACTÈRE — MOT REPRÉSENTATION DES NOMBRES DANS UN ORDINATEUR

Les éléments constituant l'ordinateur manipulent des *groupes de bits* et non pas des informations élémentaires, ou bits.

5.1 Notions de caractère

Le nombre de bits constituant le groupe le plus fréquemment présent aujourd'hui dans les ordinateurs est 8 ; le groupe de 8 bits définit l'octet.

Dans une machine à octets, les caractères (signes, chiffres, lettres) sont codés sur 8 bits.

5.2. Notion de mot

Un mot est une suite de caractères, et ce quelle que soit la nature des caractères le composant (lettres de l'alphabet, chiffres, signes). Ainsi, ce pourra être une suite d'octets.

Dans un ordinateur, le mot correspond généralement à la taille de l'information pouvant être traitée en une seule fois.

La longueur des mots est imposée par la structure des machines ; parler de machines 8 bits ou 16 bits signifie simplement que les mots traités en un seul bloc sont constitués par des groupes de 8 ou de 16 bits.

Exemples :

Le micro-ordinateur LX 500 est une machine à 8 bits.

Bit	**Binary digit = élément binaire d'information**
Digit	**Digital Unit = élément d'information numérique = un chiffre**
Mot	**ensemble de digits**
Octet	**ensemble de 8 bits**
Byte	**Le mot « byte », utilisé couramment, caractérise l'ensemble de bits formant un mot.**
Quartet	**Un ensemble de 4 bits est un quartet.**
Hextet	**L'ensemble de 6 bits est un hexet.**

Application à la notion de taille mémoire

Le préfixe K est généralement utilisé pour qualifier les tailles des mémoires ordinateurs. Ce préfixe est utilisé comme abréviation de kilo qui
— en décimal vaut 1000
— en binaire vaut 1024

Un kilobit vaut ainsi 1024 bits. Un kilo-octet représente 1024 octets.

***Exemple :** Le LX 500 a différentes tailles pour sa mémoire centrale :*
— 16 K soit 16 384 octets
— 32 K soit 32 768 octets
— 48 K soit 49 344 octets
— 64 K soit 65 536 octets

• *Toutes ces notions sont applicables aux capacités des mémoires autres que la mémoire centrale, bandes, disques, etc.*

6. ÉCRITURE EN VIRGULE FIXE, VIRGULE FLOTTANTE

Un mot de 8 bits est généralement utilisé ; une telle dimension ne permet cependant de représenter que des nombres de valeur comprise entre 0 et 255 (en faisant abstraction du bit de signe) :

• *Valeur minimum.* Tous les bits des huit positions du mot sont à 0, soit :

0000 0000

• *Valeur maximum.* Tous les bits sont à 1 ;

1111 1111

soit 255.

Or, les nombres à traiter peuvent dépasser cette valeur maximale ; il est donc nécessaire d'introduire de nouvelles conventions pour pouvoir travailler sur de plus grands nombres tout en utilisant le format à 8 bits. Ces conventions font appel à la notion de *calcul en virgule flottante.*

6.1. Notation scientifique

L'écriture d'un nombre s'appuie sur la notion de puissance. Ainsi, le nombre 1000 peut s'écrire 1×10^3. De façon générale et pour un système à base B, on pourra écrire un nombre N sous la forme

$$N = M \times B^E$$

- où M définit la mantisse de N,
- et E définit l'exposant.

6.2 Introduction du signe

Tout nombre a un signe. Celui-ci peut être *positif* ou *négatif.*

> *Un bit est suffisant pour représenter le signe.*
> *Par définition :*
>
> *0 correspond au signe +*
> *1 correspond au signe −*

Avec le signe S, on a :

$$N = S.M.B^E$$

Avec un mot de 8 bits, et en découpant les huit bits en deux ensembles de 4 bits, l'un pour la *mantisse,* l'autre pour l'*exposant,* on voit que l'on peut coder des nombres jusqu'à :

$$15 \times 2^{15}$$

soit 491 520 en décimal (voir l'exemple de la page suivante).

Le premier bit de la configuration binaire représente le signe ; ses valeurs possibles sont 0 (signe +) ou 1 (signe −). Dans ces conditions, les limites des nombres pouvant être représentés sur une configuration à 4 bits deviennent :

- *Limite inférieure :* 1111
- *Limite supérieure :* 0111

Exemple :

Configuration binaire sur 8 bits

1111	*1111*
mantisse	*exposant*

Valeur de la mantisse :
$(1111)_2 = 8+4+2+1 = 15$

Valeur de l'exposant :
$(1111)_2 = 15$

Le nombre maximum N a pour valeur :
15×2^{15}

Un mot sur un octet peut donc coder :
— en virgule fixe : -127 à 127 ;
— en notation scientifique : -10^{-40} à $+10^{+40}$.

7. CODIFICATION DES INFORMATIONS

Tous les systèmes vus jusqu'à présent sont des systèmes de numération qui seront mis en œuvre lorsque des traitements arithmétiques seront à effectuer. En réalité, l'ordinateur manipule des caractères et, pour des raisons de simplification, ceux-ci sont représentés codés.

7.1. Le décimal codé binaire ou DCB

La méthode de codage est très simple : un nombre décimal n'est pas converti globalement en binaire comme on l'a vu précédemment mais chiffre par chiffre, chaque chiffre étant représenté par 4 bits. Pour ce qui est des caractères, il suffit d'établir une correspondance avec des configurations binaires définies sur 4 positions.

Chiffres décimaux

On écrira : 0 = 0000 5 = 0101
1 = 0001 6 = 0110
2 = 0010 7 = 0111
3 = 0011 8 = 1000
4 = 0100 9 = 1001

Exemples :

• *Soit à écrire en DCB le nombre 23 : en appliquant la table de correspondance figurant ci-dessus, on obtient :*

23_{10} = *0010 0011*

• *De même, on vérifiera que 17 s'écrit*

17_{10} = *0001 0111*

Remarque :

La représentation en DCB n'a évidemment aucun rapport avec la représentation en binaire pur.

***Exemple :* 17 en binaire pur s'écrit 10001. En abrégé, le DCB se note également BCD (appellation anglo-saxonne).**

Représentation des caractères

Par caractères on entendra aussi bien les signes de l'alphabet (A à Z) que tout autre signe dont on souhaitera se doter pour s'exprimer.

Exemples :

• *caractères de ponctuation* . , (' " ? !
• *symboles de calcul* + − × /
• *symboles monétaires* $ £

Une table de correspondance fait correspondre à chaque caractère, une configuration binaire unique.

Exemples :

A = 1101
B = 1111
C = ...

Les limitations du DCB

Coder les caractères sur 4 bits introduit une limitation sur le nombre de caractères pouvant figurer dans l'alphabet complet. En effet, le nombre de configurations possibles sur la base d'une représentation sur 4 bits est de 2^4, soit 16.

Si le codage s'effectue sur 5 positions, on peut représenter 2^5 soit 32 configurations différentes : passer à 6 amène à 64 configurations. C'est cette dernière représentation sur 6 bits qui a été le plus souvent utilisée dans les ordinateurs.

7.2. L'ASCII (ou American Standard Code for Information Interchange)

L'ASCII est un code 8 bits. Le premier bit correspond au bit de parité ; les 7 autres bits servent à coder les caractères.

Le nombre de caractères pouvant être représenté dans ce système atteint 2^7, soit 128, nombre très largement suffisant puisqu'il permet de prendre en compte :

— les 26 lettres de l'alphabet (majuscules et minuscules) ;
— les 10 chiffres du système décimal ;
— et 66 signes spéciaux.

Ces signes spéciaux comprennent, en premier lieu, tous les signes de ponctuation, de calcul arithmétique, etc., usuels. Les configurations binaires restantes servent à coder d'autres signes souhaités pour l'application ou des fonctions spéciales spécifiques du périphérique utilisé.

En se reportant au tableau 1 des codes ASCII, on constate que tout caractère s'écrit sur 7 bits : soit b_7 b_6... b_1. Pour trouver l'équivalent ASCII d'un caractère, on prendra la valeur binaire b_7 b_6 b_5 correspondant à la colonne où figure ce caractère, puis la valeur binaire b_4 b_3 b_2 b_1 correspondant à la ligne. Ainsi, le chiffre 1 se traduit par 0110001, soit 31 en hexadécimal. De même, A a pour valeur 100 0001 soit 41 en hexadécimal ; l'ouverture de parenthèse « (» s'écrit 010 1000, soit 28 en hexadécimal. Le tableau 11 donne la correspondance hexadécimale pour chaque caractère.

Tableau I — Code de caractères ASCII.

b7 → b6 → b5 →						0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
BITS	b4	b3	b2	b1	Colonne / Ligne	0	1	2	3	4	5	6	7
	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
	0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
	1	0	0	1	9	HT	EM	)	9	I	Y	i	y
	1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
	1	0	1	1	11	VT	ESC	+	;	K	[	k	{
	1	1	0	0	12	FF	FS	,	<	L	\	l	|
	1	1	0	1	13	CR	GS	-	=	M	]	m	}
	1	1	1	0	14	SO	RS	.	>	N	^	n	~
	1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Nul Nul
SOH Début d'en-tête
STX Début de texte
ETX Fin de texte
EOT Fin de transmission
ENQ Interrogation
ACK Acquittement
BEL Sonnerie ou Alarme
BS Espacement arrière
HT Tabulation horizontale
LF Changement de ligne
VT Tabulation verticale
FF Alimentation papier
CR Retour de chariot
SO Changement de type de caractères
SI Changement de type de caratères
DLE Echappement de liaison de données
DCI Contrôle dispositif 1
DC2 Contrôle dispositif 2
DC3 Contrôle dispositif 3
DC4 Contrôle dispositif 4
NAF Acquittement négatif
SYN Synchronisation au repos
ETB Bloc de fin de transmission
CAN Annuler
EM Fin de support
SUB Remplacer
ESC Echappement
FS Séparateur de fichiers
GS Séparateur de groupes
RS Séparateur d'enregistrements
US Séparateur d'unités
SP Espace
DEL Effacement

Pour composer le code d'un caractère sur 7 bits, on place les bits correspondants notés b1 à b7 dans l'ordre : b7, b6, b5, b4, b3, b2, b1.

Par exemple, la lettre D se code 1000100

Tableau II — Equivalence héxadécimal, binaire et ASCII

Hexa-decimal	Binaire	ASCII
00	000 0000	NUL
01	000 0001	SOH
02	000 0010	STX
03	000 0011	ETX
04	000 0100	EOT
05	000 0101	ENQ
06	000 0110	ACK
07	000 0111	BEL
08	000 1000	BS
09	000 1001	HT
0A	000 1010	LF
0B	000 1011	VT
0C	000 1100	FF
0D	000 1101	CR
0E	000 1110	SO
0F	000 1111	SI
10	001 0000	DLE
11	001 0001	DC1
12	001 0010	DC2
13	001 0011	DC3
14	001 0100	DC4
15	001 0101	NAK
16	001 0110	SYN
17	001 0111	ETB
18	001 1000	CAN
19	001 1001	EM
1A	001 1010	SUB
1B	001 1011	ESC
1C	001 1100	FS
1D	001 1101	GS
1E	001 1110	RS
1F	001 1111	US

Hexa-decimal	Binaire	ASCII
20	010 0000	SP
21	010 0001	!
22	010 0010	"
23	010 0011	#
24	010 0100	$
25	010 0101	%
26	010 0110	&
27	010 0111	'
28	010 1000	(
29	010 1001	)
2A	010 1010	*
2B	010 1011	+
2C	010 1100	,
2D	010 1101	-
2E	010 1110	.
2F	010 1111	/
30	011 0000	0
31	011 0001	1
32	011 0010	2
33	011 0011	3
34	011 0100	4
35	011 0101	5
36	011 0110	6
37	011 0111	7
38	011 1000	8
39	011 1001	9
3A	011 1010	:
3B	011 1011	;
3C	011 1100	<
3D	011 1101	=
3E	011 1110	>
3F	011 1111	?

Tableau II (suite)

Hexa-decimal	Binaire	ASCII
40	100 0000	@
41	100 0001	A
42	100 0010	B
43	100 0011	C
44	100 0100	D
45	100 0101	E
46	100 0110	F
47	100 0111	G
48	100 1000	H
49	100 1001	I
4A	100 1010	J
4B	100 1011	K
4C	100 1100	L
4D	100 1101	M
4E	100 1110	N
4F	100 1111	O
50	101 0000	P
51	101 0001	Q
52	101 0010	R
53	101 0011	S
54	101 0100	T
55	101 0101	U
56	101 0110	V
57	101 0111	W
58	101 1000	X
59	101 1001	Y
5A	101 1010	Z
5B	101 1011	[
5C	101 1100	\
5D	101 1101	]
5E	101 1110	Λ
5F	101 1111	—

Hexa-decimal	Binaire	ASCII
60	110 0000	
61	110 0001	a
62	110 0010	b
63	110 0011	c
64	110 0100	d
65	110 0101	e
66	110 0110	f
67	110 0111	g
68	110 1000	h
69	110 1001	i
6A	110 1010	j
6B	110 1011	k
6C	110 1100	l
6D	110 1101	m
6E	110 1110	n
6F	110 1111	o
70	111 0000	p
71	111 0001	q
72	111 0010	r
73	111 0011	s
74	111 0100	t
75	111 0101	u
76	111 0110	v
77	111 0111	w
78	111 1000	x
79	111 1001	y
7A	111 1010	z
7B	111 1011	{
7C	111 1100	¦
7D	111 1101	}
7E	111 1110	~
7F	111 1111	DEL

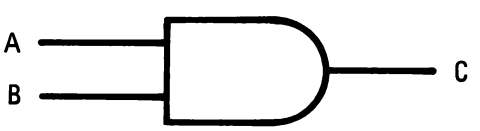
Chapitre IV

LES OPÉRATEURS LOGIQUES

*L'unité centrale d'un ordinateur et, plus précisément son UAL (*unité arithmétique et logique*) est constituée de circuits élémentaires qui vont manipuler, combiner, comparer l'information binaire.*
Les fonctions mises en œuvre peuvent être à caractère arithmétique ou logique.
Cinq circuits logiques constituent la base du fonctionnement des ordinateurs.

1. FONCTION « ET » (AND)

Le circuit qui réalise la fonction ET peut être illustré comme suit :



C prend la valeur de 1 lorsque A *et* B sont égaux à 1.

> *La conjonction* et *qui donne son nom à la fonction signifie que les conditions A = 1, B = 1 sont réalisées* simultanément.
>
> Il suffit que A ou que B soit différent de 1 pour que C soit égal à 0.

Supposons que l'on affecte à A et B une quantité binaire. Celle-ci peut prendre pour valeur 0 ou 1. A l'entrée du circuit, on peut avoir ainsi l'une des combinaisons 00, 01, 10 ou 11. L'ensemble des résultats obtenus en C pour les différentes configurations possibles pour AB peut être rassemblé dans une table de vérité.

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

La fonction ET est appelée aussi *produit logique.*
On notera celui-ci :

$$C = A \cdot B$$

2. FONCTION « OU » (OR)

Avec les mêmes conventions pour A, B et C, nous dirons que :

C = 0 si A et B sont égaux à 0 ;
C = 1, si A *ou* B est différent de 0.

Symboliquement, la fonction OU se représente comme suit :

$$C = A + B$$

Il s'agit en effet d'une *somme logique.* La table de vérité est :

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

3. FONCTION NON (NOT)

A une valeur en entrée (A), la fonction NON fait correspondre la valeur inverse (B). Ainsi :

- si A = 0, B est égal à 1
- si A = 1, B est égal à 0.

Symboliquement, on écrira $B = \overline{A}$

La table de vérité prend comme forme :

A	B
0	1
1	0

Les fonctions ET, OU et NON constituent les fonctions logiques de base ; toutes les autres fonctions logiques s'en déduisent par combinaison, aboutissant ainsi à des traitements fort complexes.

4. FONCTION « NON ET » (NAND)

Le schéma de fonctionnement d'un circuit NON ET se déduit simplement des règles auxquelles obéissent les fonctions de base ET et NON.

- ET aboutit à C = A.B.
- NON appliqué à ET implique ainsi que :

$$C = \text{NON } (A.B.)$$

ou encore : $C = \overline{A.B}$.

En d'autres termes, C est égal à 0, si, simultanément, A et B sont égaux à 1 ; C vaut 1 dans tous les autres cas.

La table de vérité devient :

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

5. FONCTION « NON OU » (NOR)

Cette fonction constitue la négation d'une fonction OU. Ainsi, si A et B sont tous deux égaux à 0, alors C vaut 1. C est égal à 0 dans tous les autres cas.

On notera : $\boxed{C = \overline{(A+B)}}$

La table de vérité devient :

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

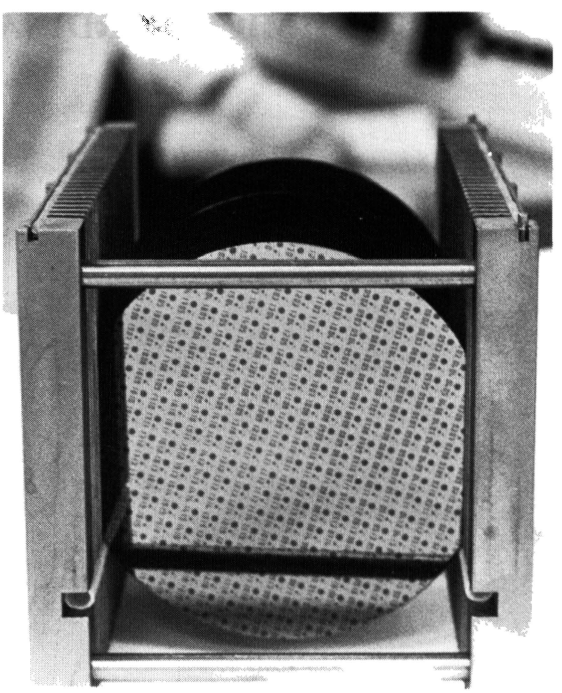
soit l'inverse de la table du OU.

A L'INTERIEUR

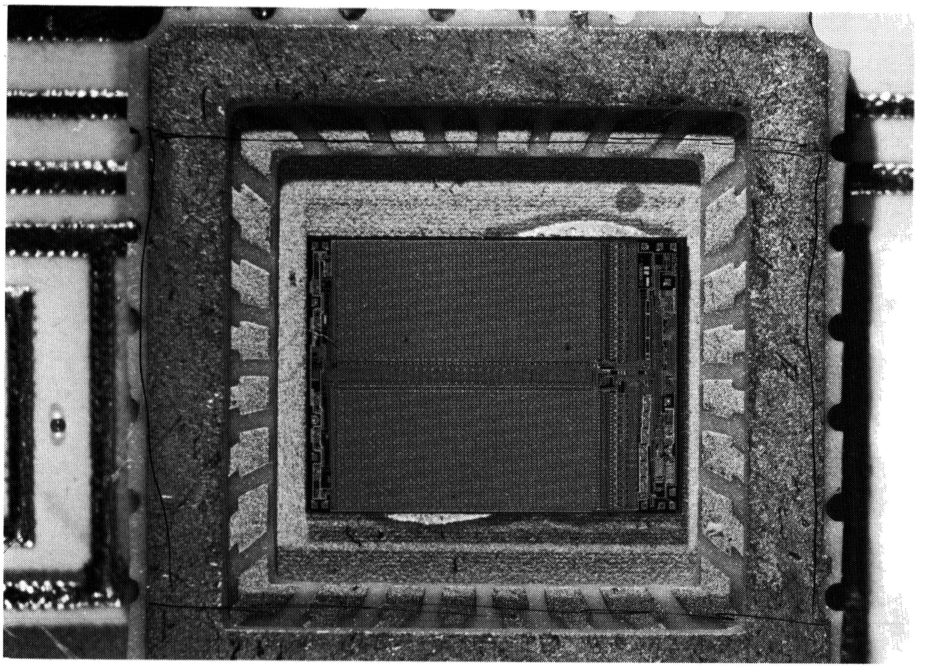
DU

MICRO-ORDINATEUR

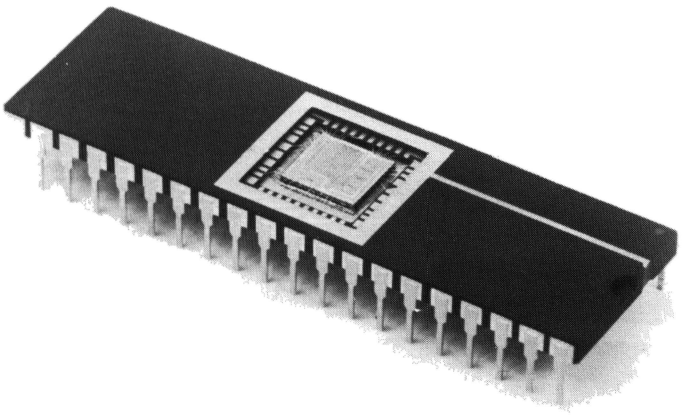
A l'origine des circuits intégrés, un lingot de silicium monocristallin, ultra-pur... *(Doc. Siemens)*



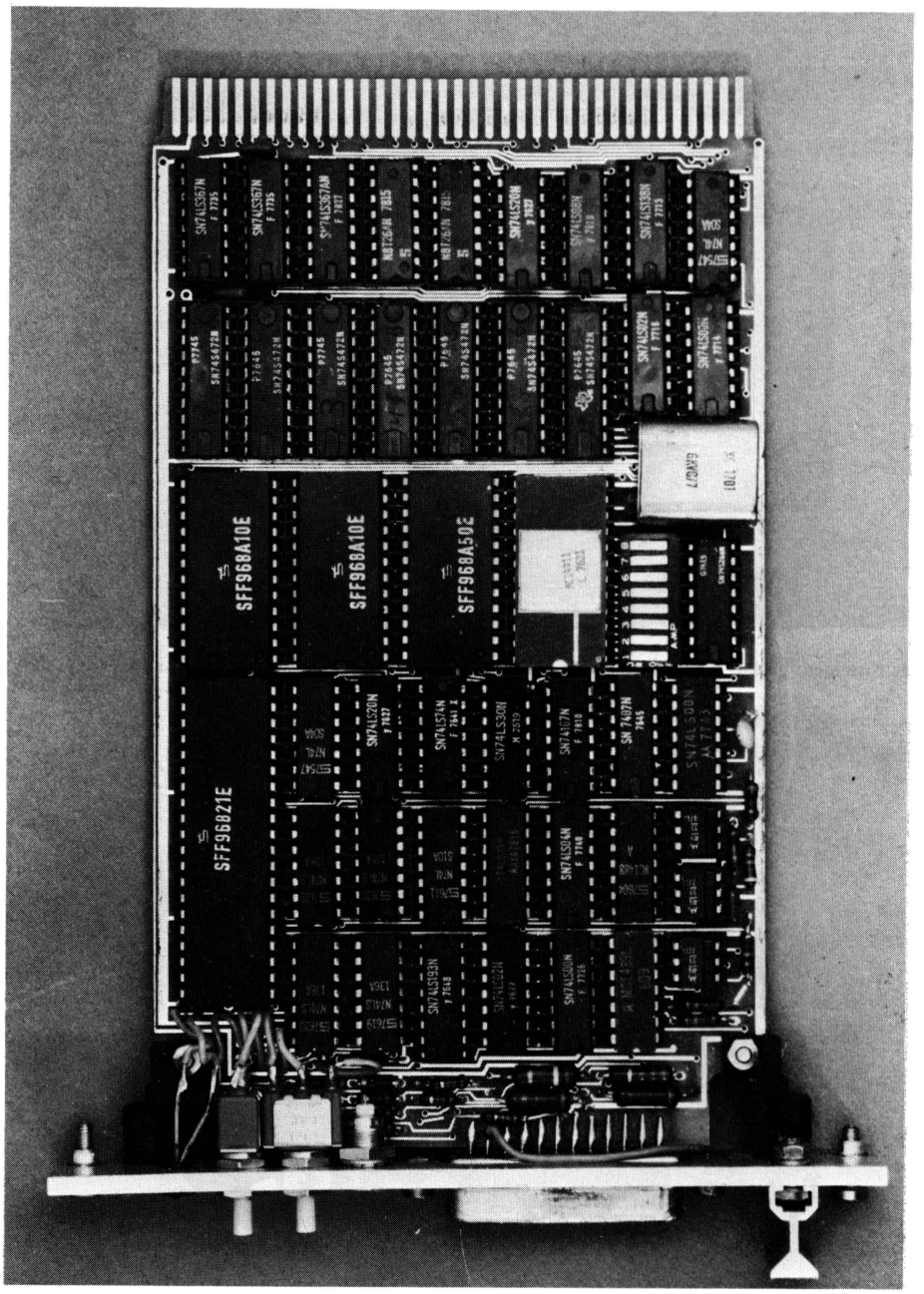
...qui sera découpé en tranches minces. Elles vont être traitées et chacune de celles-ci donnera naissance à des centaines ou des milliers de circuits intégrés. On en voit ici les motifs. *(Doc. Siemens.)*



Très fortement grossi, voici comment se présente une « puce » de circuit intégré. La surface réelle est de quelques millimètres carrés.
(Doc. Thomson-CSF - G. Warin.)



Il reste alors à fixer la « puce » de silicium dans un boîtier et à la relier aux pattes métalliques de sortie. Ce boîtier pourra ensuite être manipulé avec précautions et monté sur une carte électronique.
(Doc. Zilog.)



Une carte électronique complète, recouverte de circuits intégrés. Il s'agit ici d'une carte micro-ordinateur. ***(Doc. Gédis.)***

Chapitre V

DESCRIPTION D'UN MICRO-ORDINATEUR

1. LA MEMOIRE CENTRALE

1.1. Notion de mémoire

On appelle mémoire un dispositif capable de stocker des informations et de les restituer sur demande, et ce à n'importe quel instant. Stocker des informations revient à les ranger dans des zones bien déterminées, de façon à pouvoir les retrouver ultérieurement. Cette opération s'appelle « écriture mémoire ».

Inversement, accéder à une information préalablement stockée, implique que la zone mémoire où cette information se situe soit accessible et lisible. Accéder à une information dans une mémoire s'appelle « lecture mémoire ».

Ces deux aspects, « écriture » et « lecture », constituent deux points fondamentaux du mode de fonctionnement des mémoires.

Il est à noter que la fonction *lecture* est toujours présente dans un dispositif mémoire. La fonction *écriture*, par contre, peut très bien ne pas être permanente ; c'est le cas des mémoires dites « mémoires mortes » ou « mémoires ROM » où les informations sont inscrites une fois pour toutes et accessibles en lecture seulement.

ROM = Read Only Memory
= mémoire en lecture seulement

1.2. Organisation d'une mémoire

Les informations contenues dans une mémoire sont rangées dans des zones que, à des fins d'écriture comme de lecture, on doit pouvoir nommer séparément. Ces zones ou *surfaces de rangement* ont, par construction, un numéro qui permettra de les nommer : ce numéro est appelé « adresse ».

La vie courante donne une illustration simple de cette notion. Nous avons tous un nom, un prénom, un lieu d'habitation. Identifier notre lieu d'habitation, c'est-à-dire permettre de nous joindre, c'est trouver notre *adresse* — numéro dans la rue, nom de la rue, numéro d'arrondissement parfois, nom de la ville, etc.

Une mémoire est ainsi un ensemble de zones, chaque zone étant caractérisée par son adresse. La capacité de la mémoire est déterminée par la valeur maximale de son adresse ainsi que par la taille de chaque zone élémentaire. Chaque zone élémentaire a une taille fixe qui correspond à un mot, un mot regroupant un certain nombre d'éléments binaires. La taille d'un mot est caractéristique de l'ordinateur utilisé.

> *Pour le LX 500, par exemple, la zone fait 8 bits et sa mémoire peut atteindre la valeur maximale de 64 K avec 1 K = 1024*

1.3. Caractéristiques d'une mémoire

La capacité que nous venons de voir constitue la première des caractéristiques. Le temps d'accès et la durée du cycle de fonctionnement caractérisent également une mémoire.

Temps d'accès

C'est le temps qui sépare l'instant où on lance un ordre portant sur la mémoire et celui où l'information est disponible en sortie. Ainsi, en lecture, le temps d'accès est le temps qui sépare le moment où l'adresse de l'information à lire a été fournie et celui où cette même information sera disponible.

Cycle de fonctionnement

L'opération de lecture a pour effet, dans certaines catégories de mémoires, d'effacer l'information lue ; on dit que la lecture est *destructive.* Il faudra ainsi,

systématiquement, réécrire l'information qui vient d'être lue, opération qui prend un temps complémentaire. Le cycle de fonctionnement est ainsi supérieur au temps d'accès, de 2 à 3 fois.

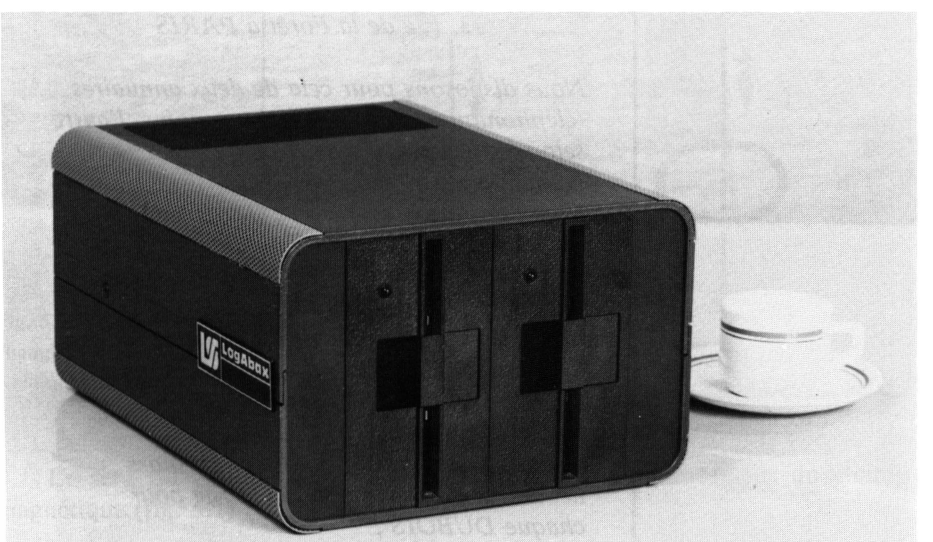
La fonction mémoire
Tout dispositif capable
- *d'enregistrer ;*
- *de conserver ;*
- *de restituer,*

une information.

1.4. Classement des mémoires

De très nombreux critères de classement peuvent être utilisés ; ces critères s'appuient sur les modes d'accès, d'utilisation, sur la capacité d'être ou non adressable.

Certains classements font également appel aux notions de temps d'accès, de coût du stockage d'un bit, de la capacité, de la rapidité de fonctionnement, etc.



L'unité centrale du LX 500 : elle comprend jusqu'à 64 K octets de mémoire.
(Doc. Logabax.)

Modes d'accès séquentiel ou direct

Accès séquentiel

Une mémoire est à *accès séquentiel* quand il est nécessaire de faire défiler, sous une tête de lecture, toutes les informations situées entre la position courante de la tête et la position où se trouve l'information désirée.

Accès direct

L'information désirée est accessible directement, à n'importe quel instant, et ce quel que soit l'endroit où elle se trouve.

Exemples :
1. En informatique :
— Accès séquentiel : bande magnétique.
— Accès direct : disque magnétique.

2. Dans la vie courante :
Soit à retrouver le numéro de téléphone d'une personne dont on connaît le nom et l'adresse :
M. DUBOIS,
3, rue de la Forêt à PARIS

Nous disposons pour cela de deux annuaires téléphoniques, l'un organisé par noms, l'autre selon les noms des rues
Annuaire par noms :
L'accès au nom est direct ; il suffit de rechercher les DUBOIS habitant rue de la Forêt.
Annuaire par rues :
L'accès au nom de la rue est également direct. Si le numéro dans la rue n'est pas connu, alors la recherche ne peut être que séquentielle, soit :
- dans l'annuaire par noms : il est alors nécessaire de lire le nom de la rue pour chaque DUBOIS ;
- dans l'annuaire par rues : il faut alors consulter l'ensemble des numéros dans la rue pour retrouver le nom.

Modes d'utilisation

Une mémoire peut être d'utilisation permanente ou temporaire ; à ce mode d'utilisation correspond généralement une technologie précise.

Adressabilité

Cette classification est très proche du classement par accès. Une mémoire à accès direct est une *mémoire adressable.* Dans les mémoires adressables, on ne travaille pas sur l'information mais sur *l'adresse* à laquelle se situe cette information.

1.5. Un peu d'histoire : la mémoire à tores

Un tore magnétique est un minuscule anneau de matière ferromagnétique. Son diamètre est de l'ordre de quelques millimètres. Traversé par un fil conducteur, cet anneau a la propriété d'être magnétisé lorsqu'un courant d'intensité I parcourt le conducteur, et de conserver cette magnétisation lorsque le courant I est supprimé.

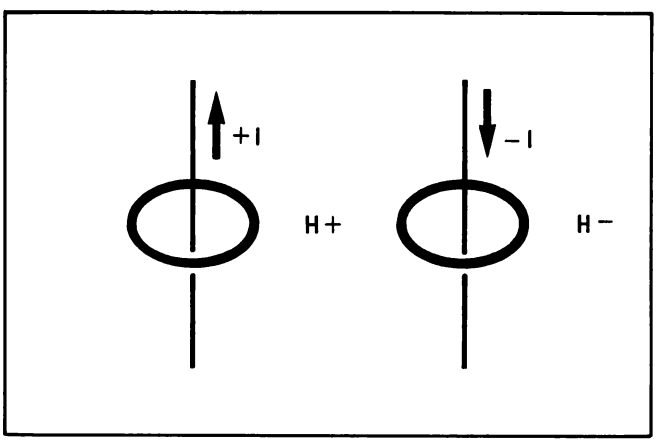


Fig. 5.1 — Création d'un champ magnétique H par un courant I.

Le sens du courant dans le fil conducteur détermine le sens du champ magnétique (fig. 5.1) :

Courant $+I \rightarrow$ Magnétisation : $+H$
Courant $-I \rightarrow$ Magnétisation : $-H$

Aux états possibles $+H$ et $-H$, on a attribué les valeurs arbitraires 1 et 0

Composition d'un tore

Supposons qu'au lieu d'un seul conducteur, l'on en ait deux, perpendiculaires entre eux, le plan qu'ils déterminent étant lui-même perpendiculaire au plan du tore (fig. 5.2). Si l'on fait passer un courant d'intensité I/2 dans les deux fils, l'intensité résultante au point de rencontre C a pour valeur I ; il y a création d'un champ magnétique +H. Si l'intensité dans chaque fil est de −I/2, on a, de même, un champ de valeur −H.

Lecture et écriture

Pour connaître l'état magnétique d'un tore, c'est-à-dire pour *lire* son contenu, il est nécessaire d'introduire la notion de *fil de détection*. Ce fil, placé parallèlement à l'un des deux fils précédents, est parcouru par un courant *induit* d'intensité I lorsque l'on introduit une variation sensible du champ magnétique dans le tore.

Opération d'écriture. Pour qu'un ferrite ait une valeur 1, il suffit de faire passer l'intensité I/2 dans les fils 1 et 2 qui la traversent.

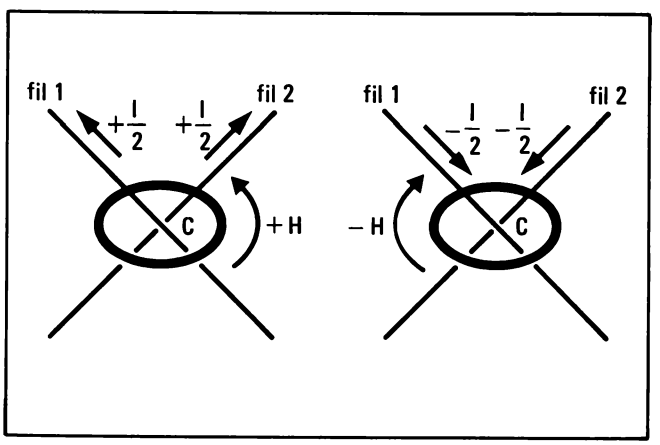


Fig. 5.2 — Action de deux courants de valeur i/2 passant à travers un tore magnétique.

Opération de lecture. Supposons que le ferrite ait pour valeur 1 ; le champ est ainsi à la valeur +H. Faisant passer un courant −I/2 dans chaque fil, nous créons un champ d'intensité −H. Le champ magnétique varie ainsi de la valeur +H à −H, induisant un courant *i* dans le fil de détection. La mesure du courant *i* dans le fil de détection permet de déterminer que la valeur initiale était 1.

Si, au départ, la valeur avait été 0, le même courant −I/2 dans chaque fil détermine un champ −H ; or, par convention, c'était là la valeur initiale du champ. Il n'y a donc pas de variation et, par conséquence, pas de courant induit.

La valeur initiale était ainsi zéro.

En résumé :
- **il y a un courant induit : la valeur lue est 1 ;**
- **il n'y a pas de courant : la valeur lue est 0.**

Constitution d'une mémoire à tores

Un tore présente deux états : il correspond à un élément binaire. Le regroupement de tores permet de constituer des mots, donc des mémoires complètes. L'opération qui consiste à faire passer des différents fils d'écriture et le fil de détection au travers des tores constitue le *tissage* de la mémoire.

1.6. Mémoires à circuits intégrés

Les mémoires à tores sont les plus anciennes ; on utilise aujourd'hui d'autres types de techniques faisant appel aux circuits intégrés. Les mémoires à circuits intégrés sont très diverses, selon qu'elles permettent la lecture seulement ou au contraire la totalité des fonctions mémoires.

La mémoire de type RAM

Les informations stockées dans les mémoires RAM peuvent être lues ou effacées pour inscrire d'autres informations à leur place. *Si l'alimentation de l'ordinateur est coupée, les informations stockées en RAM sont perdues.* Ceci peut très bien ne pas être gênant, dans le cas de résultats intermédiaires de calculs par exemple, mais à une condition seulement : le résultat final, lui, a pu être protégé, stocké de manière temporaire. L'impression sur papier, l'écriture sur un autre support comme le disque souple ou la bande magnétique constituent des moyens de sauvegarde de ces résultats. On a ainsi l'illustration de l'utilité de moyens auxiliaires de stockage ; ceux-ci seront étudiés au chapitre 6.

Mémoire de type ROM

Un autre type de mémoire — la mémoire ROM — est utilisé comme élément de la mémoire centrale dans le cas où l'on souhaite disposer de supports permanents, sans avoir toutefois besoin de faire appel à des dispositifs d'entrées/sorties onéreux ou volumineux.

Les informations ne peuvent qu'être lues ; elles sont *permanentes* dans la mesure où une quelconque action extérieure, comme par exemple le fait de ne plus alimenter l'ordinateur en énergie électrique, n'entraîne pas de perturbations.

Ces mémoires sont généralement utilisées pour stocker des programmes, en particulier celui qui contrôle le fonctionnement de l'ordinateur (programme *moniteur* ou *superviseur*). Les programmes ne sont bien évidemment stockés en mémoire ROM que s'ils satisfont aux trois conditions suivantes :

- ils sont appelés à être exécutés fréquemment ;
- ils sont exempts d'erreurs ;
- ils ne sont pas appelés à évoluer.

> ***Exemple d'utilisation des ROM :***
> Les séquences élémentaires destinées à réaliser les opérations arithmétiques ou logiques.

Le fait que les mémoires ROM ne soient utilisables qu'en lecture seulement peut être une source de gêne ; l'utilisateur lui-même peut, par ailleurs, souhaiter programmer lui-même ce type de mémoire : aussi a-t-on défini des mémoires comme les PROM, les EPROM, REPROM, etc. Ainsi :

- *Les PROM sont programmables par l'utilisateur.* Une fois programmées, elles ne peuvent être que lues.
- *Les EPROM sont des PROM qui peuvent être effacées* (par ultra-violet) *et ré-inscrites.* Ce type de mémoire est utilisé pour des programmes en cours de développement.

L'inscription d'informations dans une PROM, leur effacement si nécessaire, sont réalisés au moyen de dispositifs spécialement conçus à cet effet :

— le programmateur de PROM ;
— l'effaceur de PROM.

1.7. Technologie des mémoires à circuits intégrés

Le chapitre d'introduction à l'informatique et à l'ordinateur nous a fait faire connaissance avec les mémoires RAM, ROM, PROM, REPROM, etc. Certaines, comme les RAM, sont strictement équivalentes aux mémoires à tores de par leur aptitude à permettre la lecture, l'écriture, la conservation, l'effacement ou la ré-écriture d'informations. D'autres, comme les ROM, ne permettent que la lecture ; l'information qu'elles contiennent ne peut être modifiée. Les REPROM, quant à elles, sont des PROM autorisant l'effacement et la ré-écriture d'informations. *Toutes sont des mémoires à circuits intégrés.*

Au plan technologique, citons :
— Les procédés MOS : Metal Oxide Semiconductor.
— Les CCD = charge coupled device (ou mémoire à couplage de charge).
— Les bulles magnétiques.

Les plus fréquemment utilisées aujourd'hui sont les mémoires MOS. La fonction mémorisation est assurée de la manière suivante : une position mémoire est constituée par une bascule à deux états stables auxquels, conventionnellement, on associe les valeurs 0 et 1. Le regroupement en réseau de telles bascules constitue la *mémoire.* Cette mémoire peut être *statique* ou *dynamique :*

• *La mémoire est dite statique* car, tant que la bascule est reliée à une source d'énergie électrique, l'information qu'elle contient ne subit aucune altération.

• D'autres types de mémoires MOS présentent la particularité de se comporter comme des condensateurs. Le condensateur offre ainsi un moyen de mémorisation :

— chargé : il représente l'état 1
— non chargé : il représente l'état 0.

La charge n'étant pas constante dans le temps mais ayant tendance à diminuer (phénomène des courants de fuite), on la reconstitue périodiquement. Cette opération s'appelle le *rafraîchissement mémoire.* Ces mémoires MOS sont appelées *dynamiques.*

En résumé : la mémoire centrale

Elle permet de stocker programmes et données, que ces dernières aient été fournies à l'ordinateur par l'utilisateur, ou qu'elles résultent d'une action ou d'un ensemble d'actions de l'ordinateur lui-même.

Par définition, la mémoire doit pouvoir conserver les données.

La réalité est cependant plus complexe, la mémoire pouvant être de type permament *ou non.*

La mémoire à ferrites *ou* tores magnétiques *est de type* permanent *ou* non volatile ; *entendons par là que l'information qu'elle contient est stable, que l'ordinateur reste ou non alimenté par une source d'énergie. Cette technologie basée sur les propriétés des substances magnétiques comme l'oxyde magnétique de fer est l'une des plus anciennes :*

Les technologies utilisées aujourd'hui ne sont pas toutes permanentes ; elles peuvent être volatiles. *Les micro-ordinateurs d'aujourd'hui utilisent les deux types de mémoires :*

— des mémoires volatiles, *où l'information est présente de manière temporaire : les mémoires RAM ;*

RAM = Random Access Memory
(Mémoire à accès direct)

— des mémoires non volatiles, ou permanentes : *les mémoires ROM. Dans de telles mémoires, la configuration des bits (c'est-à-dire 0 et des 1) représentant des caractères ne change pas dans le temps. Elles ne peuvent même pas être modifiées. En d'autres termes, ces mémoires ne peuvent qu'être lues, d'où leur nom*

ROM = Read Only Memory
(Mémoire en lecture seulement)

2. FONCTIONNEMENT DE L'UNITE CENTRALE DE TRAITEMENT

2.1. Fonctions

Sans unité centrale, pas de traitement ; l'unité centrale est ainsi le cœur de l'ordinateur. Couplée à la mémoire centrale de type RAM, elle va permettre d'exécuter ce pour quoi elle a été conçue, *exécuter des programmes,* c'est-à-dire exécuter les instructions qui les composent. Or, ces instructions sont stockées en mémoire et codées.

L'unité centrale a ainsi pour rôles :
- **d'aller chercher les instructions en mémoire, une par une ;**
- **de les interpréter au moyen de ses circuits de décodage ;**
- **de s'assurer qu'elle dispose bien de toutes les informations nécessaires à son exécution ;**
- **de réaliser l'exécution proprement dite ;**
- **de mettre en réserve le résultat.**

Toutes ces opérations s'appuient sur les notions suivantes :
— adresse et compteur d'instructions ;
— le registre d'instructions ;
— l'accumulateur ;
— l'unité arithmétique et logique : UAL.

2.2. Le microprocesseur

L'ensemble de ces rôles est aujourd'hui assuré par des circuits électroniques qui constituent le *microprocesseur.* Un *microprocesseur n'est pas un micro-ordinateur ;* pour passer de l'un à l'autre il faut en effet adjoindre au microprocesseur bien des éléments : alimentation, mémoire, circuits de communication avec les dispositifs d'entrée ou de sortie, etc.

Le microprocesseur n'est pas, non plus, le seul moyen utilisable pour construire une unité centrale de traitement. A base de relais, puis de lampes et de transistors la technologie de conception des ordinateurs a connu différentes étapes en quelques dizaines d'années seulement. Chaque passage d'une technologie à l'autre est généralement considéré comme un changement de *génération* d'ordinateurs.

Le microprocesseur et, de façon plus générale, la technique des circuits intégrés constituent, aujourd'hui, le moyen le plus moderne utilisé pour bâtir des ordinateurs. Quelle que soit la technologie utilisée, le principe de fonctionnement, de même que la structure générale, restent les mêmes.

La structure s'appuie sur les notions de :
- **unité centrale ;**
- **mémoire centrale ;**
- **dispositifs d'entrée/sortie.**

Le principe de fonctionnement, quant à lui, repose sur la notion de *programme,* un programme étant constitué *d'instructions.* L'exécution des instructions, l'une après l'autre, constitue le déroulement du programme.

2.3. Les instructions

Stockées en mémoire centrale, les intructions du programme décrivent à l'ordinateur quel type d'opération est à effectuer, et sur quelles données cette opération va porter.

Le type d'opération correspond à la notion de *code opération ;* l'addition, la multiplication, la comparaison, le branchement à un endroit bien particulier, sont des opérations bien précises. A chacune d'elles correspond un code. Chaque opération travaille sur des données. Ainsi, dans le cas de l'addition, l'on sait que celle-ci porte sur deux données qui constituent les *opérandes.* Tous ces opérandes sont rangées en mémoire à des endroits bien précis appelés *adresses.*

2.4. L'exécution d'une instruction

Stockée en mémoire centrale, l'instruction à exécuter est mise à la disposition de l'unité centrale par rangement dans une zone appelée *registre instruction ;* elle y restera tout le temps nécessaire à son exécution. A ce moment, l'instruction est décodée, et le code opération est reconnu.

En fonction du type de code opération, l'unité centrale détermine quelles informations elle nécessite pour pouvoir exécuter l'instruction. Ainsi, dans le cas d'une instruction d'addition, la reconnaissance du code correspondant à l'opération d'addition déclenche la recherche de deux opérandes.

L'endroit en mémoire où se trouvent les deux opérandes figure dans l'instruction sous forme *d'adresses ;* les circuits spécialisés vont alors réaliser l'addition et ranger le résultat à l'adresse indiquée dans l'instruction.

Exemple :

L'instruction C = A + B est une instruction d'addition qui met en œuvre trois entités qui ont pour noms A, B et C.
A, B et C représentent, en réalité, des adresses de position des mémoires où sont rangées les valeurs de A et de B, et où sera rangée, après calcul, la valeur résultante.
L'exécution de l'addition consiste :

— à aller chercher à l'adresse de A, son contenu c'est-à-dire sa valeur ;
— à faire de même sur l'entité B ;
— à ajouter les deux valeurs, le résultat étant rangé à l'adresse indiquée par C.

Autres exemples :

On pourrait également imaginer une opération d'addition s'écrivant comme suit :

$$A = A + B$$

Dans ce cas, et tout en suivant le même mécanisme, le résultat de l'addition du contenu de A et de B est rangé en A, détruisant par là-même la valeur précédente contenue par A.

Enfin, des additions peuvent faire figurer des valeurs numériques explicites aux côtés d'entités.

$$C = A + 2$$

Le mécanisme est strictement identique à celui que nous venons de voir.

2.5. Enchaînement des instructions : le compteur ordinal

Nous venons de voir le mécanisme d''exécution d'une instruction. Un programme est, avons nous dit, constitué d'un ensemble d'instructions. Les opérandes figurant dans une instruction sont repérés en mémoire centrale par leurs

adresses ; il en est de même pour les instructions. L'adresse d'une instruction correspond à celle du début de la zone mémoire dans laquelle elle figure.

Pour pouvoir être retrouvée, puis amenée pour traitement dans le registre d'instruction, il est donc nécessaire que l'unité centrale connaisse cette adresse. C'est là l'objet du chargement automatique dans une zone spéciale appelée *compteur ordinal.*

Par ailleurs, une instruction possède une *longueur* bien déterminée et prend une certaine place dans la mémoire : comme les instructions d'un programme sont rangées séquentiellement en mémoire, on conçoit aisément que l'adresse de l'instruction suivante à exécuter résulte de l'addition de l'adresse de l'instruction qui vient d'être exécutée plus sa longueur.

$$\text{Adresse instruction de } n = \text{adresse } i \text{ de } n-1 + \text{longueur de } n-1$$

A tout instant, le compteur ordinal contient l'adresse de la prochaine instruction à exécuter dans le programme.

2.6. Déroulement d'un programme

Le mécanisme d'enchaînement est automatique.
L'ensemble des phases :

— lecture de l'adresse dans le compteur ordinal ;
— lecture de l'instruction en mémoire ;
— chargement dans le registre d'instructions ;
— interprétation du code opération ;
— recherche éventuelle d'opérandes en mémoire ;
— calcul proprement dit par intervention des circuits spécialisés ;
— passage à l'instruction suivante,

est répété autant de fois que nécessaire et constitue le *déroulement d'un programme.*

2.7. Eléments mis en œuvre

Le traitement des phases énumérées ci-dessus met en œuvre, outre le compteur ordinal, deux éléments (fig. 5.3) :

— *l'unité de contrôle :* elle a pour objet de traiter les informations de commande (extraire, lire, analyser, déclencher) ;
— *l'unité de traitement* qui, sur les données, exécute les traitements demandés par l'unité de contrôle.

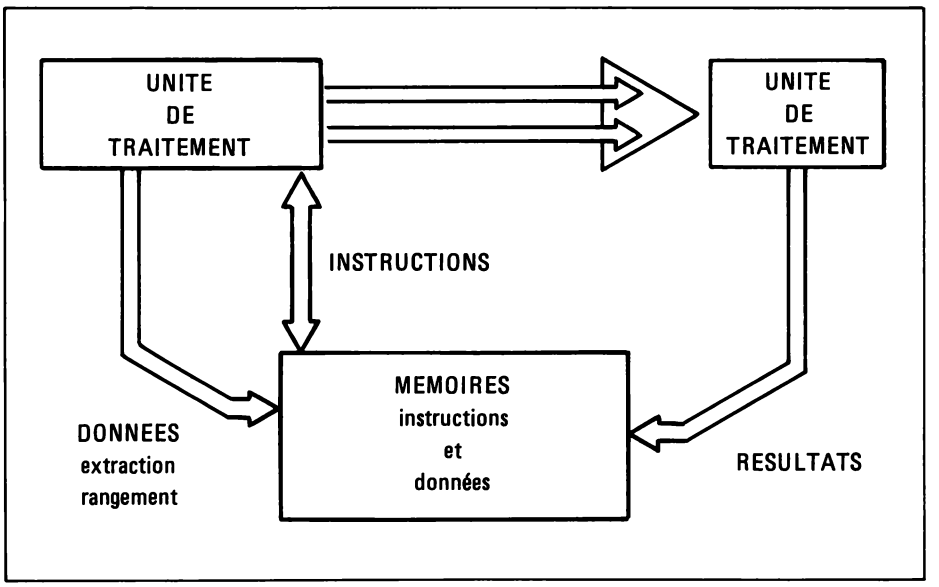


Fig. 5.3 — Circulation des informations dans un micro-ordinateur.

Registre

Un registre est une zone mémoire particulière susceptible de contenir une information. Celle-ci peut être de types divers (instructions, données) ; le stockage est temporaire.

Exemples de registres :

- *registre d'adresse ;*
- *registre d'échange ;*
- *registre instruction ;*
- *accumulateur.*

2.8. Interruption

Certains événements extérieurs peuvent entraîner l'arrêt, l'interruption de l'exécution d'un programme. Le programme en cours est interrompu, l'ordinateur commençant alors à exécuter le programme correspondant à l'événement extérieur. On dit qu'il y a eu *interruption.* Lorsque le programme voit son exécution correctement terminée, l'utilisation de l'ordinateur est rendue au programme qui a été interrompu.

Un tel mécanisme implique que certaines précautions soient prises. Notons, en particulier, la nécessité de stocker, pour pouvoir les retrouver plus tard, des informations telles que :

— le contenu du compteur ordinal ;
— des valeurs relatives au programme en cours d'exécution lorsque l'interruption survient.

On dit qu'il y a *sauvegarde* du contexte du programme interrompu.

A l'inverse, lorsque le programme correspondant à l'interruption voit son déroulement terminé, on doit retrouver les informations sauvegardées pour reprendre l'exécution du programme interrompu. On réalise alors la *restauration* du contexte.

Exemples d'interruptions

Une unité d'entrée/sortie envoie une interruption pour signaler qu'elle est en état de fonctionnement, prête à émettre ou à recevoir des informations, qu'une opération (lecture, écriture,...) est terminée, etc.

3. COMMUNICATIONS ENTRE CPU, MEMOIRES ET PERIPHERIQUES

Le fonctionnement d'un micro-ordinateur met en œuvre, en permanence, des échanges d'informations entre les divers constituants du système. Il est donc nécessaire de prévoir, au stade de la construction du micro-ordinateur, des voies de transit de ces informations ; ces voies sont souvent appelées *BUS* (fig. 5.4).

L'information de base connue et traitée par l'ordinateur est non pas constituée par le *bit* mais par un ensemble de bits, l'ensemble caractérisant l'ordinateur. Les ensembles à 8 bits sont aujourd'hui les plus répandus. Il n'est pas pensable de transférer ces bits un à un ; aussi les transfère-t-on par groupes, de 8 bits en général, en s'appuyant sur des bus constitués de 8 conducteurs. Chaque conducteur travaille sur le bit ; les extrémités du bus ne connaissent par contre que des groupes de 8 bits.

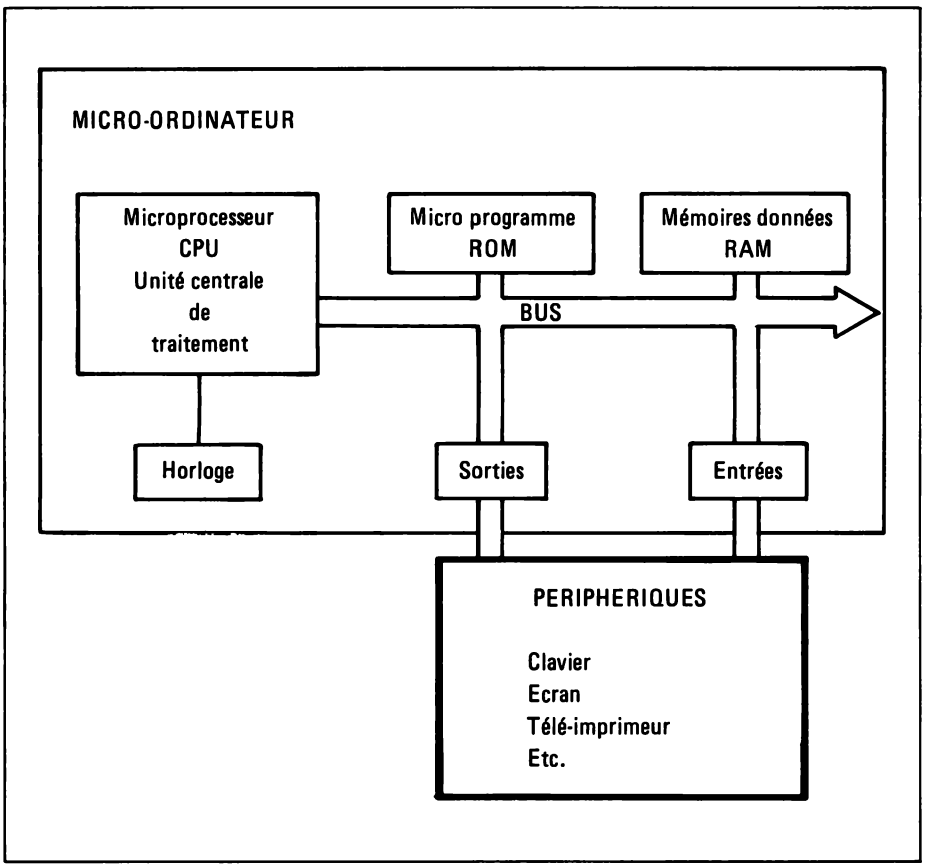


Fig. 5.4 — Conception générale d'un micro-ordinateur.

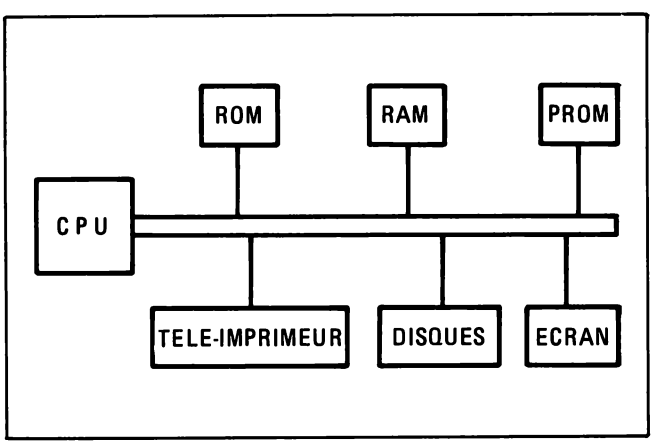


Fig. 5.5 — Structure à bus unique.

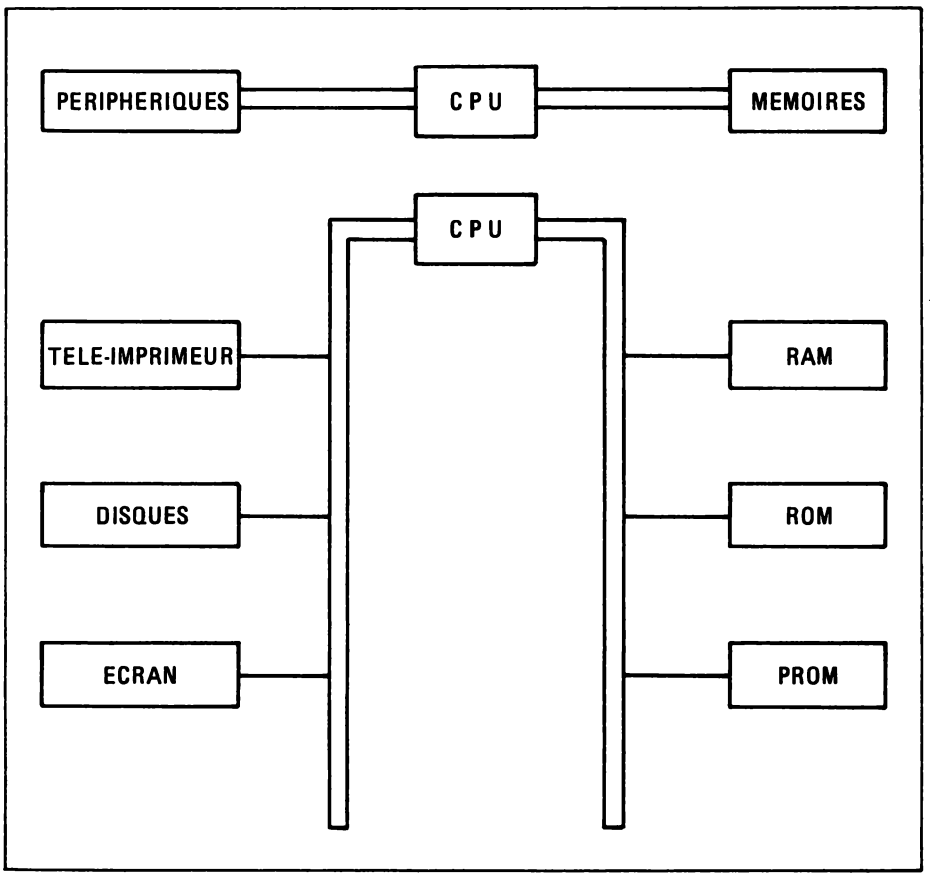


Fig. 5.6 — Structure à deux bus.

Le type des informations peut, par ailleurs, être différencié en *adresses* de données et les *données* proprement dites :

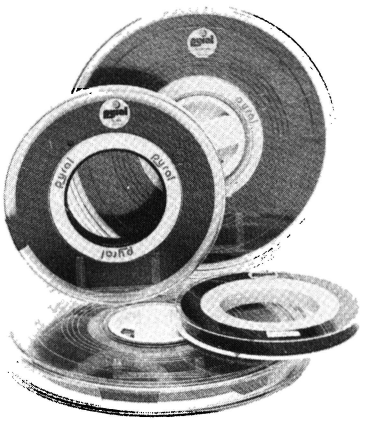
- Un bus qui ne véhicule que des *adresses* est un *bus adresse.*
- Un bus qui ne véhicule que des données est un *bus de données.*
- L'un et l'autre sont de types *parallèles.*

Le nombre de bus dans un micro-ordinateur dépend de l'architecture même du micro-ordinateur. Les schémas figurant ci-dessus donnent des exemples de structures à un et à deux bus (fig. 5.5 et 5.6)

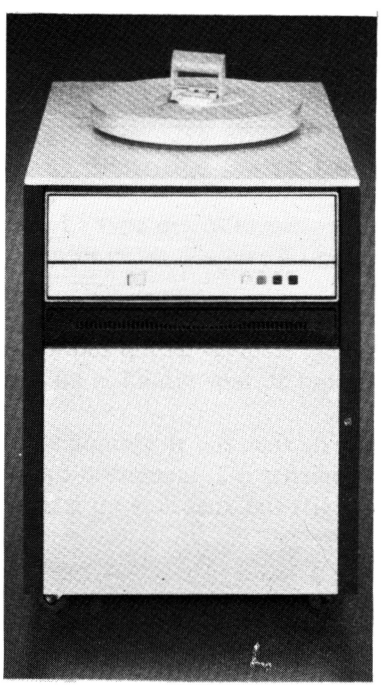
AUTOUR

DU

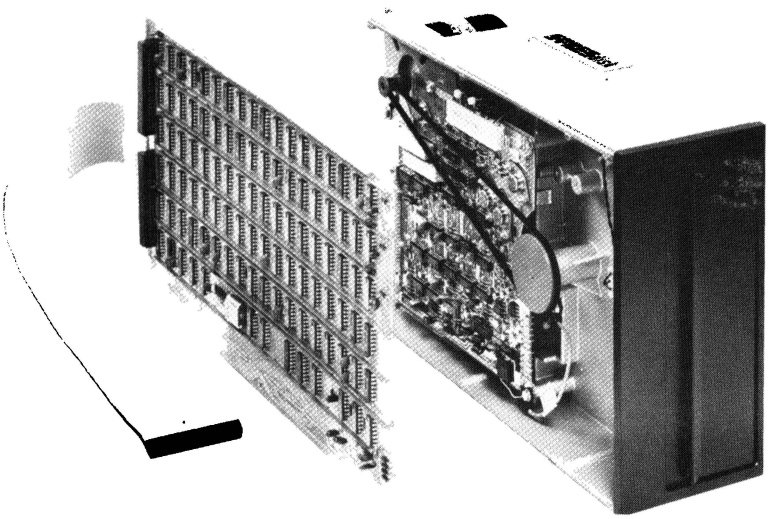
MICRO-ORDINATEUR



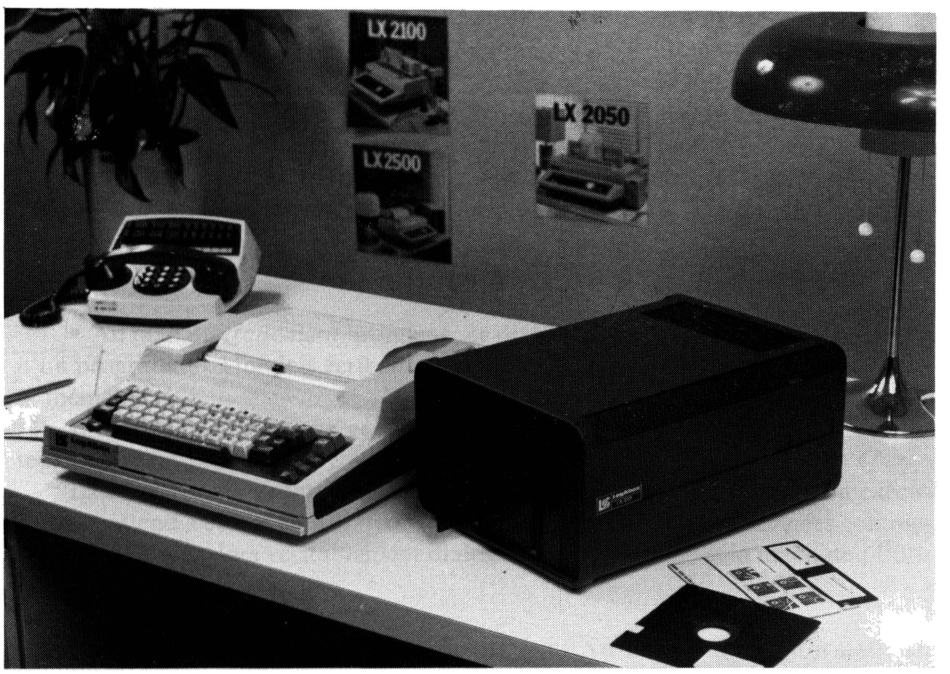
Les bandes magnétiques servent de mémoires périphériques. Leur présentation en cassettes rend leur manipulation plus aisée (ci-dessous). ***(Doc. Pyral.)***



Unité à disques magnétiques. Les disques magnétiques offrent une grande capacité de stockage et le temps d'accès à l'information est réduit. ***(Doc. Logabax.)***



Unité à disquette, avec sa mécanique d'entraînement et son électronique d'accompagnement vue ouverte. ***(Doc. Sykes.)***



Poste de travail complet LX 500 Logabax avec micro-ordinateur et double unité à disquette, et imprimante. ***(Doc. Logabax.)***

Chapitre VI

LES PÉRIPHÉRIQUES

1. PÉRIPHÉRIQUES À VISUALISATION

L'unité centrale regroupe, comme on l'a vu, l'ensemble des fonctions de traitement arithmétique et logique de l'ordinateur. L'utilisateur d'un ordinateur peut souhaiter, sinon suivre pas à pas le déroulement de l'exécution d'un programme (ce qui est d'ailleurs possible), du moins connaître certaines valeurs intermédiaires ou finales d'un traitement. Par ailleurs, il peut avoir besoin de disposer d'un mécanisme d'entrée d'informations (valeurs, ordres, etc...) pour traduire, au niveau de l'ordinateur, quel type d'opération il désire réaliser.

Ceci est rendu possible par deux types de dialogues avec la machine :

• Au plan du *traitement* lui-même, ce dialogue est traduit par l'intermédiaire d'un programme. Une fois écrit et mis au point, le programme est enregistré dans la mémoire.

L'exécution n'implique généralement pas d'action particulière de l'utilisateur ; il est dit *passif*.

La structure du programme peut cependant être telle que durant son exécution il s'attend à ce que l'utilisateur fournisse des précisions, des données complémentaires. Il y a alors véritablement dialogue entre l'homme et la machine ; l'utilisateur est dit *actif*.

• Au plan du *pilotage* du traitement, et au vu de certaines informations, l'utilisateur peut décider de réaliser une séquence particulière de traitement plutôt qu'une autre. Il oriente alors le traitement en fournissant à l'ordinateur des ordres ou commandes auxquels l'ordinateur va réagir.

Pour s'exprimer, l'utilisateur emploie un langage spécifique dit *langage de commande* ou *langage de dialogue.*

RELATION HOMME-MACHINE

Attitude passive
• *Ordinateur + programme enregistré.*
• *Programme exprimé dans un langage de programmation.*
• *Participation de l'homme :*
— nulle au niveau de la programmation ; celle-ci est déjà réalisée ;
— faible au niveau de l'opérateur.

Attitude semi-active
• *Ordinateur + programme enregistré*
• *Présence d'un dialogue prévu dans le programme comme la demande d'instructions de données, la réponse à des questions, etc...*
• *Utilisateur et opérateur correspondent.*

Attitude active
• *Programmation inter-active,*
ou :
• *déroulement du ou des programmes sous la responsabilité de l'utilisateur.*
• *Utilisateur et opérateur sont uniques.*

1.1. Communication « homme-machine »

Qui dit *dialogue* implique *communication.* La communication « homme-machine » doit obéir à un ensemble de règles :

• Le langage d'expression doit être *commun.* Ne pas connaître une langue étrangère interdit, par exemple dans le cas de l'anglais, de communiquer avec un sujet britannique.

• Ce langage doit avoir un *support ;* pour communiquer avec autrui, divers supports sont possibles :

— par écrit ; le support est le *papier ;*

— par la voix ; le support est l'*ouïe* qui transmet les vibrations produites par les sons. L'oreille humaine se charge de la réception, le cerveau en assurant, compte tenu des connaissances acquises, la reconnaissance et l'interprétation ;

— par la vue ; le support est l'*image* que, comme dans le cas de la voix, des dispositifs particuliers permettent de recevoir et d'interpréter.

Tous ces supports ont leurs équivalents en informatique, et à chaque support correspond un moyen mécanique, électrique ou électronique de réception et d'interprétation :

Type de Communication	**Support**	**Equivalent informatique**
Ecriture	Papier	Imprimante
	Image	Ecran
Son		Synthétiseur - dispositif à reconnaissance vocale

1.2. La visualisation d'informations

L'écran/clavier constitue le dispositif de visualisation le plus répandu aux côtés de l'imprimante. Cette dernière fournit un moyen *permanent* de visualisation ; une fois imprimé, le papier, la trace écrite restent tels quels. L'écran fournit une trace temporaire.

Les deux fonctions de dialogue : entrée et sortie de données, sont assurées par deux dispositifs distincts :

Entrée de données - le clavier
Ce dispositif permet d'introduire des données utiles au traitement. Ces données peuvent être des lettres, des chiffres, des mots, des ordres.

Sortie des données : l'écran
Il permet de visualiser les informations introduites, les résultats obtenus, partiels ou non, et, de façon générale, toute information utile à la compréhension du déroulement de l'application.

1.3. Le tube à rayons cathodiques

C'est la technique la plus fréquemment utilisée. Dans le domaine des équipements grand public, elle est parfaitement illustrée par le téléviseur présent dans les foyers. Au plan informatique, on voit ce type d'écran de plus en plus fréquemment dans des organismes au contact du grand public comme les compagnies aériennes, les professions du voyages, la S.N.C.F., les banques, etc...

Les informations visualisées sont de type :
— alphabétique (lettres A à Z) ;
— numérique (chifffres 0 à 9) ;
— spécial (opérateurs arithmétiques + − × par exemple).

Si l'écran ne permet que *l'affichage* de caractères alphabétiques (au sens large : les lettres de l'alphabet plus un certain nombre de caractères spéciaux) et numériques, l'écran est dit alphanumérique. Si, en outre, l'écran permet de visualiser des dessins, l'écran est dit *graphique.*

1.4. Principe des écrans cathodiques

Les images visualisées sur un écran cathodique — on parle aussi de *tube à rayons cathodiques* ou *CRT (cathode ray tube)* — sont constituées au moyen d'un faisceau d'électrons qui, par l'intermédiaire de dispositifs de concentration et de déviation, atteint un écran sur lequel est disposée une matière luminescente (luminophore). La composition de cette matière détermine la couleur finale des images.

Les électrons sont produits à partir d'une cathode par chauffage. Chauffée, la cathode a en effet la propriété de perdre un certain nombre d'électrons — ce phénomène est connu sous le nom d'*émission thermo-ionique.* L'ensemble des électrons constitue un faisceau que l'on va faire converger vers l'écran. Cette convergence est assurée au moyen de dispositifs de concentration constitués de lentilles électrostatiques ou électromagnétiques.

Un faisceau est d'autant plus étroit que la convergence est mieux réalisée. L'image donnée sur l'écran par l'impact du faisceau sur le luminophore s'appelle *spot :* c'est un point. Ce spot, ou point lumineux, va alors être déplacé selon des schémas prédéfinis, pour constituer l'image des chiffres, lettres, etc., que l'on désire visualiser.

1.5. Constitution d'un écran

On peut distinguer quatre ensembles principaux :

Le tube. Il regroupe tous les éléments nécessaires à la visualisation : source d'émission d'électrons ; dispositifs de concentration, d'accélération, de déviation ; écran avec une matière luminescente.

Entretien de l'image. L'image est créée lors du choc qui se produit entre le faisceau d'électrons et la matière luminescente qui tapisse la face intérieure de l'écran. Cette image n'est pas permanente ; elle n'est pas « persistante ». La rémanence maximum d'un luminophore n'excède pas quelques secondes. Pour conserver une image fixe sur l'écran, il est donc nécessaire de la ré-émettre à intervalles réguliers. Cette action a ainsi pour objet *d'entretenir l'image ;* on parle aussi de régénération ou de *rafraîchissement.* L'opération d'entretien de l'image fait appel à un dispositif qui dispose, dans sa propre mémoire, de l'information préalablement inscrite.

L'électronique de commande. Elle a pour rôle d'assurer l'enchaînement logique des opérations nécessaires à la constitution de l'image et à son entretien.

L'interface ordinateur.

1.6. Affichage des caractères

Les dispositifs de déviation permettent au faisceau de couvrir l'ensemble de l'écran et donc d'accéder à tous les éléments qui le constituent. Cette opération est appelée *balayage.*

Les balayages peuvent être de plusieurs types :

• l'écran est balayé de façon circulaire ; c'est le cas par exemple des écrans radars. Le balayage est dit *circulaire.*

• l'écran est balayé ligne à ligne, l'image étant constituée de lignes parallèles. Les points constituant chaque ligne sont plus ou moins lumineux.

Dans ce cas le balayage est dit *balayage type télévision.*

• On peut souhaiter, enfin, n'accéder qu'aux points véritablement utiles dans la constitution d'une image. Le balayage repose alors sur des positionnements successifs aux endroits utiles ; il s'agit alors d'un *balayage cavalier.*

Le balayage — circulaire, télévision ou cavalier — caractérise ainsi la manière dont le faisceau d'électrons explore l'écran de visualisation. La visualisa-

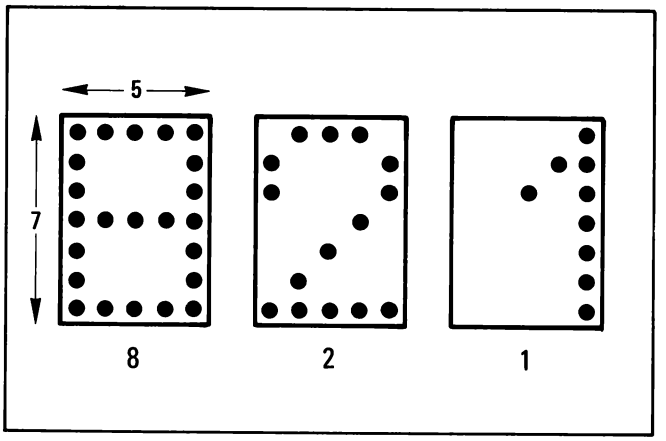


Fig. 6-1. Exemples de chiffres réalisés sur une matrice de 5 par 7 points.

tion proprement dite s'appuie, dans le cas des terminaux alphanumériques, sur une construction par points.

Généralement, le caractère s'inscrit dans une matrice de 5×7 ou 7×9 points. La figure 6-1 en donne un exemple.

1.7. Types de terminaux à écran cathodique

Le téléviseur traditionnel, par définition, dispose des fonctions de génération et de visualisation d'images ; il est donc utilisable comme dispositif d'affichage par des ordinateurs.

Malgré son prix attrayant dû aux grandes séries produites pour la consommation de masse, le téléviseur domestique n'est cependant pas à recommander pour utilisation par un ordinateur, ne serait-ce que pour des raisons de confort de l'utilisateur (lisibilité des caractères, fatigue visuelle...)

Le terminal à écran vidéo ou écran professionnel, bien que plus onéreux, est la solution la plus utilisée aujourd'hui. L'adjonction d'un clavier (possible aussi sur un téléviseur ordinaire) lui fournit la possibilité de communiquer avec l'ordinateur.

1.8. Autres technologies

- *Technologie à plasma :* mise à profit de la luminescence d'un gaz ionisé.
- *Technologie des cristaux liquides :* utilisation de l'état intermédiaire liquide-solide de certains corps organiques.
- *Technologie holographique.*

2. LE CLAVIER

Sa fonction principale est de servir de moyen de transmission de l'information ; il se présente sous la forme d'un certain nombre de touches dont l'actionnement entraîne la transmission d'une configuration binaire particulière à l'écran ou à l'ordinateur.

L'exemple le plus courant d'un clavier est fourni par celui de la machine à écrire. A chaque touche correspond, par construction, un symbole particulier, chiffre, lettre ou caractère spécial.

Dans une machine à écrire mécanique, appuyer sur la touche A, par exemple, entraîne la sélection de la tige métallique qui porte, gravé, ce même caractère. L'impression proprement dite résulte de la transmission de la pression fournie par le doigt sur la touche à l'ensemble d'impression - porte-caractère/ruban/papier.

Les claviers utilisés sur les terminaux à écran utilisent un principe semblable. Il ne s'agit pas cependant de transmettre une pression mais plutôt un signal électrique, caractéristique de la touche enfoncée, dont le décodage conduira à la détermination précise du caractère dont il s'agit et au lancement des opérations de reconstitution de ce caractère au moyen du faisceau d'électrons.

Pour faciliter le travail de l'utilisateur, les claviers sont généralement construits dans le souci de rassembler, en un même lieu, des familles de caractères, voire même des fonctions complètes. Dans sa version française, le clavier offre une distribution des lettres appelée AZERTY, du nom des premiers caractères alphabétiques de la première rangée. D'autres types de claviers, QWERTY, QWERTZ, sont en usage dans les pays anglo-saxons.

2.1. Composition d'un clavier alphanumérique

En standard, le clavier alphanumérique regroupe : les lettres de l'alphabet de A à Z, les chiffres de 0 à 9, ainsi que divers caractères tels que les symboles arithmétiques usuels +, −, ×, /, les symboles usuels de comparaison égal à (=), différent de (≠), supérieur à (>), inférieur à (<) ; les symboles de ponctuation (, ; ? !), etc.

Divers autres symboles peuvent bien entendu être définis au niveau du clavier ; leur utilisation spécialise alors le terminal.

Exemples de touches spéciales

- *Terminal scientifique :*
  — *présence de touches exponentielles, logarithme, racine carrée, etc.*
- *Terminal de calcul APL :*
  — *opérateurs spéciaux.*

2.2 Touches fonctions

Utilisé dans le cas d'une application bien précise, un opérateur de terminal peut faire appel très fréquemment à des séquences d'instructions qui réalisent des fonctions bien précises et pré-déterminées. *Ces séquences sont de véritables programmes.* Pour des raisons de gain de temps et de confort au poste de travail, ces programmes sont associés à des touches. L'enfoncement d'une touche équivaut alors à l'appel du programme qui lui correspond et au lancement de son exécution.

2.3 Touches numériques

Bien que présents au niveau du clavier standard, les chiffres figurent souvent une deuxième fois dans les claviers. Ils peuvent être intégrés au clavier standard ou, au contraire, séparés.

Ce sont là des options généralement proposées par les constructeurs dans un souci de mieux-être pour l'opérateur du terminal.

2.4. Fonctions complémentaires : la gestion du curseur

Afin de rendre l'utilisation d'un écran/clavier plus aisée, des fonctions complémentaires sont généralement proposées, dont la gestion du curseur.

La visualisation est faite de gauche à droite de l'écran sur une même ligne, puis ligne à ligne. A chaque instant, un symbole identifie la position dans laquelle s'inscrira le prochain caractère introduit. Ce symbole a pour nom *curseur ;* il est souvent matérialisé par un petit carré lumineux ou par un simple trait (souligné). Il peut être *clignotant.*

Une fois rempli, un écran se présente sous forme de lignes avec des pleins (mots, chiffres, etc..) et des vides. On peut souhaiter vouloir remplir certaines zones ou en modifier d'autres. Ceci implique que l'on sache se positionner sur une zone bien précise de l'écran. Cette facilité s'appuie sur des possibilités de déplacement du curseur dans les quatre directions : bas, haut, droite et gauche.

Exemple de position du curseur :

APPLICATION CALCUL
X = 1
*RESULTAT Y = X**2*
Y = 1
INTRODUCTION nouvelle valeur ?
(oui/non)

1re position d'écriture libre

Parmi les autres possibilités, on notera les clignotements, la surbrillance, l'inversion vidéo.

3. PÉRIPHÉRIQUES ÉLECTRO-MAGNÉTIQUES

Toute information, pour être traitée, doit être présente en mémoire centrale ainsi qu'il a été vu précédemment. Quelle que soit sa nature — donnée ou programme — cette information a un caractère *transitoire* qui résulte de la volatilité du type de mémoire utilisée pour constituer les mémoires centrales.

C'est là le revers d'une technologie pourtant fort attrayante à bien des égards, son temps d'accès en particulier. Par ailleurs, et bien que l'on connaisse des décroissances de coût spectaculaires, la mémoire centrale est généralement chère, trop chère pour conserver des informations dont le volume peut être rapidement important.

Pour concilier les diverses exigences de l'utilisateur - temps de traitement le plus court possible, volume de stockage important, stabilité de l'information dans le temps (non volatilité des mémoires), l'on a été ainsi amené à concevoir de nouveaux types de mémoires en faisant appel aux propriétés magnétiques de certaines substances.

Les dispositifs permettant de stocker l'information sur supports magnétiques sont appelés « périphériques électromagnétiques » ; ils constituent des mémoires auxiliaires ou secondaires pour l'ordinateur et répondent aux divers critères techniques et économiques requis.

3.1 Périphériques et supports

L'information peut être stockée sur une grande variété de supports : badge, compte à pistes, bandes, disques magnétiques, disques souples, etc... La lecture et l'écriture des informations sont, entre autres fonctions, réalisées par les dispositifs spéciaux que sont les périphériques électromagnétiques.

Un exemple simple permet d'illustrer ces deux notions ; il s'agit du magnétophone à cassettes grand public si largement répandu aujourd'hui. La cassette, qu'elle soit enregistrée ou non, constitue le *support*. Le dispositif permettant lecture, enregistrement, etc... est constitué par le *magnétophone* proprement dit. En ce sens, le magnétophone constitue un *périphérique de lecture et d'enregistrement.*

Revenons un instant sur la notion de support. Celui-ci présente différentes caractéristiques :

- Une fois enregistrée, et sauf accident ou manipulation volontaire, la musique ou la parole restent enregistrées. La bande magnétique ou la cassette sont ainsi des supports *non volatils.*

• Accéder à une portion d'enregistrement située, par exemple, au tiers de la bande implique un déroulement de la bande de son début au point précis où commence le morceau choisi.

En d'autres termes, ce type de support n'autorise qu'un accès progressif, *séquentiel.*

Si par contre, nous avions pris comme exemple le disque, nous aurions été amenés à définir un *accès direct.* En effet, l'accès à un morceau peut se faire directement par positionnement du bras de lecture sur le début du morceau choisi.

Accès *séquentiel,* accès *direct,* constituent deux notions fondamentales en informatique. A l'une comme à l'autre correspondent des types de périphériques comme des modes d'exploitation précis.

• Sauf accident ou manipulation volontaire, l'information est et reste, nous l'avons dit, enregistrée. L'accident ? il s'agit par exemple de l'appui sur une touche effacement (lorsqu'elle existe), ou tout simplement d'un enregistrement qui se superpose à un enregistrement déjà effectué. Pour les cassettes préenregistrées, cet accident ne saurait se produire car, par construction, un dispositif généralement mécanique protège la cassette des enregistrements ou effacements intempestifs.

La notion sous-jacente a pour nom *« protection écriture ».*

La manipulation volontaire ? C'est là le cas courant de l'utilisation d'un appareil à dicter. Une fois transcrit, le texte enregistré n'a plus aucune valeur et la bande peut être ré-utilisée. Cette possibilité de ré-utilisation est particulièrement intéressante, compte tenu du coût du stockage.

• La cassette de notre exemple est enfin *amovible ;* entendons par là la possibilité de changer de support tout en utilisant le même dispositif de lecture/écriture.

3.2. Constitution du support : types différents

Pour les exemples qui nous intéressent ici, il s'agit dans tous les cas de supports à base de matières plastiques sur lesquels un oxyde magnétique a été

déposé. Parmi les types les plus répandus, nous citerons la *bande,* la *cassette* (qui n'est en réalité qu'une petite bande magnétique), le *disque dur* et le *disque souple.*

Pour les micro-ordinateurs, la cassette a d'abord été le support mémoire auxiliaire le plus fréquemment utilisé. Le disque souple est aujourd'hui le moyen le plus utilisé en raison de ses qualités aux plans de l'utilisation et de la fiabilité.

3.3. Bande magnétique

Type de support : bande

Constitution :

- Support polyester
- Enduction d'un seul côté
- Couche d'oxyde magnétique dispersée dans un liant

Caractéristiques :

— stabilité mécanique ;
— mémoire non volatile ;
— accès séquentiel ;
— largeur courante : 12,7 mm (0,5 pouce) ;
— longueur très variable : 90 m à 970 m ;
— grande capacité dépendant de la densité d'enregistrement.
Exemple : plus de 100 millions de caractères sur une bande de 730 m.

— 7 ou 9 pistes.

3.4. Cassette magnétique

Comparable dans son principe à la bande magnétique, utilisable dans les mêmes conditions, la cassette magnétique représente le moyen de stockage auxiliaire le moins cher qui existe actuellement. C'est là la raison essentielle du succès de ce type de support, auprès des fabricants de petits systèmes tout d'abord (facturières, ordinateurs de bureau, outils de saisie) et des fournisseurs de micro-ordinateurs ensuite.

La cassette a cependant longtemps souffert d'un problème de fiabilité ; l'évolution de la technologie disques souples (augmentation de la capacité, fiabilité, accès direct) a fait qu'aujourd'hui, ce type de support est de moins en moins utilisé sur les systèmes dits professionnels. Les produits grand public la font cependant figurer encore sur leurs catalogues pour des raisons de prix essentiellement.

Les principales caractéristiques de la cassette numérique compacte généralement utilisée sont :

- Norme : ECMA
- largeur : 3,81 mm
- longueur : 90 m
- densité d'enregistrement : 800 bpi (bpi : bits par inch - 1 inch : 25,4 mm)
- vitesse de recherche : 3 m/s (120 ips) (ips : inches par seconde)
- vitesse d'enregistrement : 0,5 m/s (20 ips).

3.5. Disques

La couche d'oxyde magnétique est déposée sur un plateau circulaire en aluminium sur une épaisseur d'environ 5 microns. L'opération de dépôt constitue ce que les fournisseurs appellent la phase d'*enduction ;* elle peut porter sur une face seulement ou sur les deux faces du disque. L'information figure sur des pistes concentriques, l'accès se faisant au moyen d'un ensemble de têtes de lecture ou d'écriture que l'on positionne précisément au début de la piste.

Physiquement, c'est-à-dire extérieurement, le disque utilisé en informatique ressemble aux disques courant utilisés en audio. Pour être lu, le disque est placé sur un dispositif dont les principales fonctions sont :

— la mise en rotation du disque ;
— le positionnement de l'ensemble des têtes (*).

Ce dispositif — platine de lecture en audio — a pour équivalent en informatique le lecteur/enregistreur de disques magnétiques *(disk drive,* dans la littérature anglo-saxonne). Là s'arrête l'analogie avec le disque audio.

Disque musical	**Disque informatique**
• 1 piste par face en hélice	• Plusieurs pistes concentriques
• Lecture seulement Accès direct par voie manuelle	• Lecture/Ecriture • Accès direct automatique
• 1 tête de lecture	• Plusieurs têtes de lecture ou d'enregistrement

(*) Certains types de disques possèdent des ensembles de têtes fixes, à raison d'une tête par piste présente sur le disque. L'opération de positionnement n'a plus, dans ce cas, de raison d'être.

Principaux types de disques

Deux grandes catégories peuvent être distinguées :

— *les unités dites à têtes fixes,* généralement utilisées comme mémoires auxiliaires de mini-systèmes pour applications dites de temps réel ;
— *les unités à bras mobile ;* porteur de têtes, ce bras a pour fonction l'exploration de l'intégralité de la surface porteuse d'informations.

Disques à têtes fixes

En général, le support n'est pas amovible. Ce type de disque peut être caractérisé par :
— une capacité moyenne : de 1 à 80 millions de bits ;
— un temps d'accès moyen de l'ordre de 10 ms ;
— une vitesse de transfert de plusieurs mégahertz, de 1 à 10 MHz.

Ils sont généralement utilisés pour des applications nécessitant des temps d'accès très courts (contrôle de processus industriels, applications temps réel telles le contrôle de trajectoires balistiques, commutation électronique de messages ou de circuits...)

Disques à bras mobiles

Le support est alors amovible ; il comprend le plus souvent plusieurs disques empilés l'un au-dessus de l'autre, chaque face pouvant être explorée par une tête. Dans de telles piles de disques, le mécanisme d'accès à l'information est ainsi

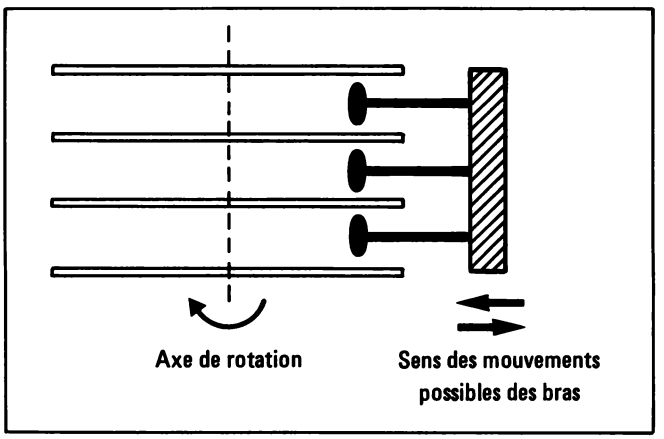


Fig. 6-2. Le bloc des bras mobiles portant les têtes de lectures pour pile de disques.

constitué de plusieurs bras, chaque bras portant deux têtes ; l'une est destinée à l'exploration de la face inférieure du disque supérieur, l'autre à celle de la face supérieure du disque inférieur.

Le schéma de la figure 6-2 illustre la notion de « *peignes* » de têtes. Plusieurs types de disques amovibles sont actuellement présents sur le marché :

• **Les cartouches**

Un cartouche est un simple disque dont les deux faces sont utilisées pour le stockage des informations. L'introduction dans le dispositif de lecture ou d'enregistrement peut se faire de deux manières, soit horizontalement auquel cas une fente est pratiquée dans l'unité de lecture de disques (à l'image des « mange-disques » grand public), ou verticalement par le haut de l'appareil.

• **Les piles de disques**

L'ensemble est constitué de plusieurs disques solidaires d'un même axe ; des empilements comptant jusqu'à 12 disques ne sont pas rares.

Les faces extérieures des disques supérieur et inférieur, les plus exposées car en contact avec le milieu extérieur (poussières, sources de chaleur, pollution par contact, etc...) ne sont pas fréquemment utilisées. L'introduction dans l'unité de disques se fait verticalement.

• **Disques type « INFOPAC »**

Il s'agit là de produits intermédiaires dans lesquels la qualité d'amovibilité est conférée à l'ensemble support-têtes, celui-ci étant placé dans une enceinte hermétique. L'avantage immédiat au plan de la fiabilité réside dans l'isolation du support et des têtes, tous deux éléments fort fragiles, vis-à-vis du milieu extérieur.

3.6. Organisation des informations sur les disques

Si l'accès aux informations portées par un support magnétique peut être strictement séquentiel (à l'image de l'accès sur bandes ou cassettes magnétiques), le disque, de par son organisation, offre l'avantage essentiel de permettre un *accès direct.*

L'information est, sur chaque face, organisée en pistes et en secteurs. A chaque secteur correspond une capacité exprimée en caractères, une piste regroupant un nombre déterminé de secteurs.

Placé sur un plateau, le disque est mis en rotation lors de la mise sous tension de l'unité de disques. Dans les dispositifs dits à têtes fixes — une tête par piste en général — l'ensemble des informations situées sur une même piste passe ainsi

A titre d'exemple, le disque type 5440 dont la capacité totale est de 5 millions d'octets a une organisation qui s'appuie sur la décomposition suivante :
- *chaque piste comprend 24 secteurs ;*
- *813 pistes ;*
- *un secteur correspond à 256 caractères.*

sous la tête correspondant à la piste. Pour les dispositifs à têtes mobiles, la lecture d'une information nécessite le déplacement de l'ensemble tête-support de tête jusqu'à la piste concernant l'information ; le positionnement étant effectué, on se trouve dans le cas précédent.

Notion de cylindre

Certains types de supports disques utilisent, on l'a vu, des empilements de disques. La notion de piste se retrouve bien entendu d'un disque à l'autre dans la pile. L'ensemble des pistes situées sur une même verticale constitue un *cylindre* (fig. 6-3).

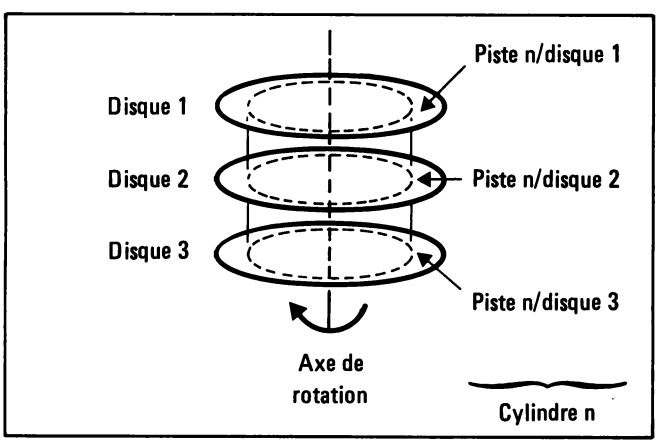


Fig. 6-3. Cylindre de disques.

Notion de temps d'accès

Nous supposerons tout d'abord que le disque, mis en rotation, a atteint la vitesse de rotation uniforme ; cette opération, qui prend un temps de l'ordre de quelques minutes, est réalisée, une fois pour toutes, lors de la mise sous tension ou lors de toute installation de disque ou de pile de disques. La vitesse est de l'ordre de plusieurs milliers de tours par minute.

Toute information est caractérisée par des valeurs numériques qui indiquent dans quel secteur, ou sur quelle piste ou quel cylindre, avec quel numéro d'ordre sur la piste (des informations différentes peuvent être en effet en séquence sur la piste) elle est située.

L'ensemble de ces informations constitue l'*adresse* de l'information. Lire ou écrire une information d'adresse bien déterminée, entraîne les opérations automatiques suivantes :

- fourniture de l'adresse à l'unité de disques ;
- décodage de cette adresse en :
  - — numéro de secteur ;
  - — numéro de piste ou de cylindre ;
  - — numéro d'ordre ;
- positionnement de la tête ou du peigne de têtes sur la piste ou le cylindre concerné ;
- détection du numéro d'ordre ;
- lecture ou écriture.

Le *temps d'accès* correspond à la somme des temps partiels nécessaires à la réalisation des différentes opérations que nous venons d'énumérer. Un ordre de grandeur de temps d'accès :

— Unité à bras mobile : 30 à 40 ms (ms : milliseconde)
— Unité à têtes fixes : 10 ms

- *Vitesse de rotation :* elle est de 1500 à 3600 tours par minute.
- *Distance tête - support :* elle est de quelques millionièmes de mètres (μm).

3.7. Disque souple

Le disque souple, ou *disquette,* reprend les caractéristiques principales du disque, aussi bien sur le plan de sa constitution que sur celui de son utilisation. Le support de la couche magnétisable est en nylon ; il est placé dans une pochette de forme carrée en vinyl ou en carton souple. L'ensemble est très léger et non rigide, d'où son nom.

Ce type de support est apparu au début des années 1970 sur des équipements destinés à des applications de saisie (c'est-à-dire d'enregistrement) des données (*IBM,* 1973). Offrant des possibilités d'accès direct sur un support flexible de capacité raisonnable, ce moyen de stockage a connu, depuis lors, diverses étapes dont les principales sont les suivantes :

- *Le disque souple 8'' ou disquette :*

D'un diamètre de 8 pouces (soit 25,4 mm $\times$ 8 = 203 mm) il offre dès son annonce une possibilité de stockage de 256 K caractères (avec 1 K = 1024). L'information est inscrite sur une seule face ; la densité d'écriture, c'est-à-dire le nombre d'informations élémentaires par pouce, est de 3200 bpi (bpi = *bits par inch ou par pouce).*

La disquette comprend 77 pistes ce qui, compte tenu de son diamètre, aboutit à une densité radiale (ou transversale) de 48 pistes par pouce (unité tpi = *track per inch*). Chaque piste est découpée en secteurs ; un secteur regroupant un certain nombre d'octets.

Le format le plus universel aujourd'hui correspond à :

— 77 pistes ;
— 26 secteurs par piste ;
— 128 octets par secteur.

• *le disque souple 5 1/4" ou mini-disquette :*

D'un format inférieur et, par conséquent, d'une capacité inférieure, de l'ordre de 90 K octets, la mini-disquette apparaît vers 1975. Présentant les mêmes caractéristiques d'utilisation que son aînée, la mini-disquette est aujourd'hui le moyen de stockage auxiliaire le plus répandu sur les micro-ordinateurs.

• *Vers le double face*

C'est là le premier pas effectué par les constructeurs vers une augmentation de la capacité sans pour autant changer les caractéristiques dimensionnelles du support. Utiliser le double-face, c'est, d'emblée, doubler la capacité. Cette évolution, sur le disque souple comme sur la mini-disquette, se fera durant les années 76-77.

• *Vers des densités plus élevées*

Sur une surface bien déterminée, augmenter la capacité globale revient à rechercher le moyen d'augmenter le nombre d'informations contenues par une piste, et/ou à augmenter le nombre de pistes par face :

— agir au niveau de la capacité revient à augmenter la *densité longitudinale* (unité bpi) ;
— agir au niveau du nombre de pistes s'exprime en termes d'augmentation de la *densité radiale* (unité tpi).

L'option double densité est maintenant des plus fréquentes au niveau des disques ou mini-disques souples. Elle conduit à des capacités de
• 512 K pour la disquette
• 200 K pour la mini-disquette

Les options simultanées - double face/double densité conduisent, quant à elles, à des capacités de 400 à 1000 K.

Caractéristiques typiques de disques souples

— Disquette SAGEM (modèle DS 3) :

Capacité par disque : 3.1 M bits
Capacité par piste : 40800 bits
Densité d'enregistrement : 3200 bpi
Densité de piste : 48 tpi
Nombre de pistes : 77
Temps d'accès à l'information :
piste à piste : 10 ms
stabilisation : 10 ms
Vitesse de rotation : 360 tr/mn ± 3 %
Durée d'un demi-tour : 83 ms
Enregistrement : double fréquence
Type de tête : écriture/lecture avec effacement
Protection d'écriture : Oui
Le disque comporte 77 pistes concentriques espacées de 0,53 mm sur une seule face (48 pistes par pouce).

(Source : documentation SAGEM)

3.8. La disquette

Support magnétique, flexible, la disquette est fragile. Sa manipulation requiert les mêmes précautions que les autres types de supports - disques, bandes, etc... C'est là la raison d'être de l'existence d'une enveloppe scellée, le tout constituant véritablement le disque souple. En examinant un disque dans son enveloppe, on peut relever la présence de plusieurs orifices :

• Un orifice circulaire est destiné à laisser place à un dispositif qui, par pression, maintiendra le disque bien placé. Sa mise en rotation assurera la rotation du support à l'intérieur de son enveloppe protectrice.

• La lecture et l'enregistrement sont réalisés au moyen de têtes ; le principe est strictement le même que pour les disques. L'orifice réservé à cet effet permet aux têtes de venir au contact du support.

• Le disque souple comprend un nombre bien déterminé de secteurs. Le repérage des secteurs ou celui des pistes met en œuvre ce que l'on appelle un *index,* le disque lui-même étant dit *sectorisé.* La sectorisation peut être réalisée physiquement (on fait alors des trous dans le support, et ces trous sont repérés au moyen de dispositifs photoélectriques) ou par comptage à partir d'une référence dénommée « signal d'index ».

• On peut souhaiter n'utiliser un support tel le disque souple qu'en lecture seulement. A cet effet, on collera une pastille sur l'encoche réservée à cet effet. L'information qu'il supporte est ainsi permanente, l'accès en écriture étant interdit. Cette interdiction met en œuvre un dispositif de *protection écriture.*

TABLEAU DE QUELQUES UNITES A DISQUETTES DISPONIBLES EN FRANCE

Modèle	Capacité (mégabits)	Vitesse de transfert (K bits/s)	Temps d'accès (ms) à la piste n	Nombre de pistes	Densité (bpi)	Vitesse de rotation (tr/mn)	Compatibilité avec IBM
IBM 3740	3.1	250	80 + 50 n	77	3200	360	oui
Control Data 9400	3,1	240	10 + 10 n	77	3200	360	oui
Calcomp 140	3,2	250	10 + 6 n	77	3200	360	oui
Shugart 800/801	3,2/6,4	250/500	83 + 8 n	77	3200/6400	360	oui
Potter 4740	3,1	250	35 n	77	3270	360	oui
Logabax 45	3,1	250	17,5 + 2,5n	77	3270	360	oui
Sagem DS3	3,1	250	10 + 10 n	77	3200	360	oui
Memorex 651	2,5	250	10 + 10 n	64	3100	375	non
Memorex 652	3,2	250	10 + 10 n	77	3270	360	oui
DRI 74	3,1	250	10 + n	77	3270	360	oui

(Source : Mini-Micros)

3.9. Unité à disques souples et caractéristiques

Comme tout dispositif de lecture et d'enregistrement, l'unité à disque souple comprend les éléments nécessaires à :

— L'introduction du support : il s'agit d'une simple fente ; celle-ci peut être verticale ou horizontale selon la disposition de l'unité.

— Sa fixation dans l'unité.

— La mise en rotation du disque : moteur plus dispositif d'entraînement, plus dispositif de contrôle.

— L'ensemble tête-support de tête.

— L'électronique de logique et de commande.

— La protection de l'ensemble, contre des poussières en particulier-enceinte et porte.

Mémoires comparées

Type	Adressage des données	Temps d'accès moyen	Volume moyen stocké (en octets)
Unité de stockage (type 3850)	direct	—	50 à 300 milliards
Bande magnétique	séquentiel	15 s	10 millions
Disque amovible	direct	50 ms	20 à 600 millions
Disque fixe	direct	10 ms	0,5 à 20 millions
Disque souple	direct	400 ms	250.000
Cassette	séquentiel	10 s	100.000

4. PRÉCAUTIONS À PRENDRE DANS LA MANIPULATION DES SUPPORTS MAGNÉTIQUES

Les supports magnétiques, comme leur nom l'indique, utilisent les propriétés magnétiques de conservation ou d'effacement de corps tels les oxydes magnétiques. Ils sont constitués :

— du support proprement dit (plastique, métallique, etc...) ;

— d'une fine couche magnétique, elle-même constituée d'une matière magnétisable ou démagnétisable et d'un liant.

Les fonctions de lecture, d'écriture, d'effacement sont assurées au moyen de dispositifs électro-mécaniques qui, dans le cadre de leur fonctionnement, font appel à des techniques électroniques. Ces dispositifs ne sont pas, en général, en contact avec le support, mais à une distance extrêmement réduite.

L'espace entre la tête ou l'ensemble de têtes est des plus réduits - de quelques dizaines de microns à quelques microns ; on le dénomme, en général, *hauteur de vol.*

L'ensemble de ces caractéristiques permet d'établir la liste des éléments extérieurs qui, hors précautions, peuvent conduire à une détérioration de l'ensemble, voire même à une destruction de l'information. Ces éléments sont : *la chaleur, les poussières, les champs magnétiques, les chocs.* Ils peuvent conduire à une détérioration momentanée ou permanente, partielle ou totale.

5. SORTIE PAPIER

5.1. La fonction impression

Calculer, choisir, trier sur la base de méthodes mathématiques ou logiques, voilà la base même de l'utilité des ordinateurs.

Le résultat du calcul, du choix, du tri, est un objectif que se définit l'utilisateur ; qu'il soit final ou au contraire, partiel, c'est-à-dire point de départ d'un autre raisonnement mathématique ou logique, ce résultat doit être connu, mieux visualisé, dans certains cas même, conservé. Cette conservation est à la base d'une industrie de la « *trace écrite* » par opposition à la « *trace fugace* ». L'*imprimante* est le dispositif qui permet d'obtenir cette trace écrite ; l'écran de visualisation, fournira, lui, une trace fugace.

5.2. Les imprimantes

L'imprimante la plus simple, et aujourd'hui encore la plus répandue, est la *machine à écrire ;* convenablement modifiée, la machine à écrire classique a été en effet le premier dispositif imprimant à être utilisé en appoint d'une unité de calcul. Les machines comptables, les facturières et autres dispositifs destinés à des tâches de gestion courante dans les entreprises (paie, comptabilité, facturation) en étaient dotés, il n'y a pas encore si longtemps.

Le télé-imprimeur avec son cylindre d'impression portant quatre rangée de caractères constitue l'un des plus anciens exemples de dispositif imprimant utilisé à la périphérie de l'ordinateur.

L'un et l'autre vont nous permettre d'introduire quelques notions dont la prise en compte progressive conduira aux différentes technologies d'impression. Tous deux fournissent une trace papier ; cette trace peut être conservée, voire emportée. Les auteurs anglo-saxons parlent de « *hard copy* », une *copie dure.*

Les *performances* enregistrées de ces dispositifs, c'est-à-dire le nombre de caractères pouvant être imprimés dans une durée de temps bien déterminée, sont de l'ordre de 10 caractères par seconde.

L'insuffisance notoire de ces performances, eu égard aux volumes à imprimer en sortie d'ordinateur, est l'une des raisons qui ont conduit à chercher d'autres types de moyens d'impression.

5.3. Différents types d'imprimantes

Deux grandes familles peuvent être distinguées : à impact ou « non impact ».

Les imprimantes à impact

Mettant en jeu des procédés mécaniques ou électromécaniques, le mécanisme des imprimantes à impact repose toujours sur le principe d'une *frappe,* par un marteau, d'un élément dont la mise en action aboutira à l'impression partielle ou totale d'un caractère.

L'imprimante *à aiguilles* est l'exemple typique d'une impression partielle. Ce n'est que progressivement, par points, que le caractère est constitué, édité.

La machine à cylindres (type télé-imprimeur), la machine à écrire standard (un marteau par caractère), les machines à marguerite, à corolle, etc... sont des exemples de dispositifs imprimants à caractères *pré-formés.* L'action de frappe conduit à l'impression d'un caractère complet.

Imprimantes à marguerite

L'ensemble des caractères imprimables réside sur une petite roue dont chaque extrémité de rayon porte l'image en relief d'un caractère. L'analogie avec une *marguerite,* chaque pétale étant considéré comme support d'un caractère, est immédiate.

Une marguerite possède généralement un jeu de 128 caractères ; l'accès à d'autres jeux de caractères est rendu possible par *l'amovibilité* de la marguerite, à l'image de ce qui se fait sur des machines de secrétariat (boule, sphère, etc.).

Ce type de machine est généralement utilisé dans les applications où la qualité de l'impression est particulièrement recherchée (frappe de travaux de secrétariat tels courrier direction, etc.). Par contre, les performances sont relativement modestes (de l'ordre de 30 à 60 caractères par seconde).
Principe d'impression. L'impression d'un caractère requiert les opérations suivantes : positionnement du système d'impression sur la ligne ; rotation du disque marguerite, Badmington, etc..., de façon à amener le caractère à imprimer au

regard d'un marteau ; mise en action du marteau ; impression dûe à la pression de l'ensemble marteau/caractère sur la feuille de papier au travers d'un ruban encreur.

Imprimantes à aiguilles

Les imprimantes à aiguilles sont également des imprimantes à impact. Leurs performances sont très largement supérieures à celles des imprimantes type marguerite ; celles-ci culminent cependant à 2 ou 300 caractères par seconde. Au-delà, il est nécessaire de faire appel à des technologies différentes, imprimantes à bandes par exemple. On peut, dans ces conditions, obtenir des vitesses d'impression qui ne se mesurent plus en caractères mais en dizaines de lignes par seconde (jusqu'à 2000 lignes/minute).

Ce type d'imprimantes ne sera pas décrit ici, les performances pouvant être atteintes étant d'un tout ordre de grandeur que le débit de l'ordinateur qui l'alimente. Des considérations financières conduisent également à écarter ce type d'imprimantes sur des configurations micro-ordinateurs.

Une configuration typique de micro-ordinateur comprend une imprimante basée sur la technologie à aiguilles, ou sur la technologie à marguerite.

Imprimantes sans impact

Les techniques utilisées sont fort diverses. L'impression thermique, électrostatique, à laser, à jet d'encre en constituent quelques exemples. Les performances sont fort variables, de quelques dizaines de caractères par seconde à plus de 18 000 lignes par minute.

Classification succincte des imprimantes

— Très basse vitesse : < 30 cps (caractères par seconde)
— Basse vitesse : 30 à 100 cps
— Moyenne vitesse : 300 à 1000 lpm (lignes par minute)
— Grande vitesse : 1000 à 3000 lpm
— Très grande vitesse : > 3000 lpm

5.4. Imprimantes à aiguilles

Basées sur un principe relativement ancien puisque datant de plus d'un siècle, les imprimantes à aiguilles ne feront véritablement leur apparition en quantités sur le marché qu'au début des années 1970. Si le principe est simple, sa mise en application est en effet beaucoup plus complexe, la simplicité des produits n'étant qu'apparente.

Quel en est le principe ? Les caractères à imprimer ne le sont pas en une seule action, mais progressivement. A chaque caractère, alphabétique, numérique ou

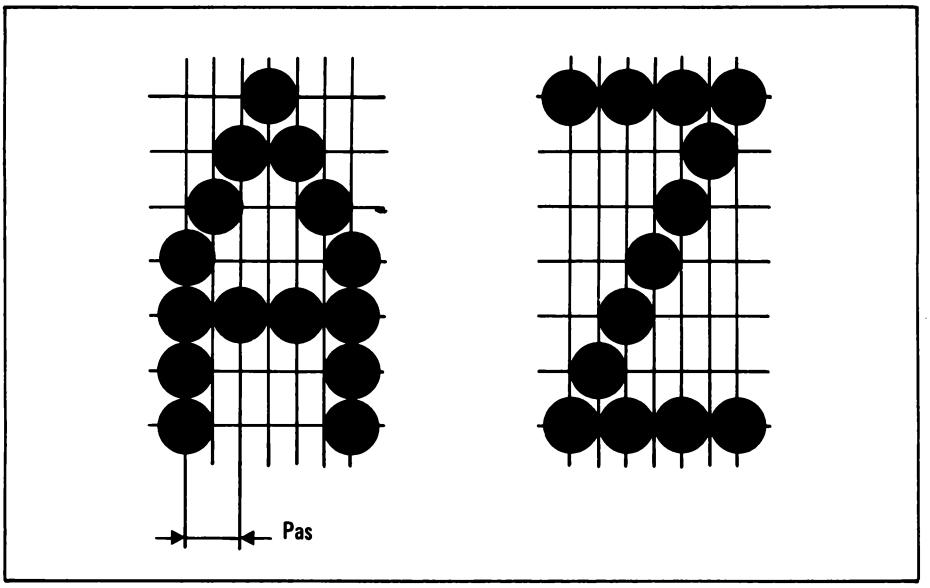


Fig. 6-4. La grille de définition du graphisme est composée de 7 colonnes verticales et 7 lignes horizontales ; il en résulte 7 emplacements possibles par ligne pour la frappe d'un point. Par construction, l'espace minimal séparant les centres de 2 frappes successives sur une ligne est égal au double de la distance entre deux colonnes.

spécial, correspond une configuration bien précise de points. Cette notion de configuration est illustrée par la figure 6-4, laquelle représente le caractère alphabétique A.

Regardons attentivement cette configuration ; nous constatons que le caractère A est constitué d'un certain nombre de points disposés sur une *grille,* leur disposition relative étant caractéristique du caractère. Chaque point correspond à la frappe sur le papier d'une aiguille.

La première colonne comprenant les points a été constituée par la mise en jeu simultanée de 4 aiguilles. Poursuivant notre examen, nous constatons qu'il y a un seul point sur la 2^{e} colonne, 2 points sur la 3^{e}, etc...

Pour la clarté de l'explication, les lignes et les colonnes sur lesquelles se situent ces points ont été visualisées sur la figure ; dans la réalité, bien évidemment, seuls apparaissent les points imprimés par l'action des aiguilles. Cette grille a aussi l'avantage d'expliquer la raison pour lesquelles une telle technique d'impression est appelée « *impression mosaïque* » ou « *impression matricielle* ».

Elle permet enfin d'introduire la notion de « définition » du caractère. Notre caractère A s'inscrit dans une grille constituée de 7 lignes et 7 colonnes. Le nombre de lignes fournit la définition verticale, le nombre de colonnes, la définition horizontale.

On conçoit aisément que multiplier le nombre de lignes ou de colonnes amène à avoir *une meilleure définition,* en d'autres termes une meilleure qualité de l'impression.

Les matrices les plus largement utilisées sont les matrices 7×7 ou 7×9, mais cependant, rien ne s'oppose à ce que des définitions plus précises soient utilisées si ce n'est une complexité croissante. Ainsi sont progressivement constitués, par frappes successives des colonnes de points, les caractères de l'alphabet, les chiffres et les caractères spéciaux utilisés.

L'organe d'impression utilisé, ou *tête d'impression,* rassemble les divers moyens nécessaires à la frappe. Quels sont ces moyens (en nous limitant aux principaux d'entre eux) ? Les voici :

— *Les aiguilles :* placées sur une même verticale, leur nombre définit la qualité qu'aura l'impression.

— *Les électro-aimants :* à raison d'un par aiguille, ils actionnent l'aiguille dont la frappe donnera lieu à l'impression d'un point.

— *Les guides-aiguilles.*

L'ensemble de l'organe d'impression se déplace parallèlement à la feuille de papier le long d'un guide mécanique, le mouvement lui-même étant déterminé par un moteur à entraînement continu.

Principales qualités d'une imprimante à aiguilles

Trois éléments sont généralement considérés comme caractéristiques d'une imprimante :

• *La qualité de l'impression :* celle-ci sera d'autant meilleure que le nombre d'aiguilles sera grand et l'espace inter-aiguilles petit, de façon à donner, bien qu'avec une trace discontinue, l'image d'un caractère plein.

• *La vitesse :* pour ce type de technologie, la vitesse atteint facilement 200 caractères par seconde.

• *Le contrôle du mouvement du papier :* lorsqu'une ligne est entièrement terminée, l'organe d'impression revient à sa position de repos, à l'extrême gauche, cependant que le papier effectue automatiquement un saut de ligne.

A ces éléments de base s'ajoutent un certain nombre de facilités, d'options qui confèrent à l'imprimante sa souplesse d'utilisation : types différents d'*entraînement papier ; possibilité d'utiliser du papier sous forme de liasses continues* (en pliage accordéon ou en rouleau par exemple) ou d'éléments discontinus. Dans ce dernier cas, divers *introducteurs* sont associés à l'imprimante.

Imprimantes mono et multi-têtes

Le principe d'impression par aiguilles, très largement répandu aujourd'hui, peut s'appliquer différemment :

a - la tête d'impression peut être unique et elle se déplace à vitesse constante le long du papier ;

b - l'imprimante peut posséder plusieurs têtes d'impression, solidaires d'un même mécanisme ;

c - l'imprimante peut utiliser un peigne d'aiguilles, chaque aiguille ayant, lors de ses oscillations contrôlées par la logique de l'imprimante, la possibilité de frapper autour d'une position moyenne.

Imprimantes uni ou bidirectionnelles

Dans les dispositifs imprimants les plus simples, l'impression se fait caractère par caractère, de la gauche vers la droite. Lorsque la dernière position d'écriture est atteinte sur la ligne à imprimer, la tête est automatiquement ramenée à sa position de repos cependant que le papier est mû de façon à effectuer un saut de ligne. Cette brève description nous permet d'introduire plusieurs notions :

— l'imprimante est du type *caractère ;*

— l'impression est *unidirectionnelle.*

Certaines imprimantes mettent à profit le retour même — on parle aussi de *retour chariot* — pour imprimer également de la droite vers la gauche. Dans ce cas, l'imprimante est dite *bidirectionnelle :* elle imprime dans les deux sens.

5.5. Jeux et générateurs de caractères

Une imprimante est d'autant plus riche en possibilités d'impression que son jeu est complet. On appelle *jeu de caractères* l'ensemble des lettres, chiffres ou symboles imprimables. Généralement, un jeu comprend :

— les lettres de l'alphabet A, B, C, ... Y, Z ;

— les nombres décimaux : 0, 1. 2, ... 8, 9 ;

— divers symboles spéciaux tels que +, −, ×, / etc...

Ceci, tant en majuscules qu'en minuscules. Plus le jeu de caractères est complet, plus l'impression est riche, plus le texte imprimé est varié, agréable à lire, clair. A ces diverses possibilités qui, d'ailleurs, reprennent intégralement celles offertes sur une machine à écrire, s'en ajoutent d'autres telles que :

— écriture dilatée ;

— écriture penchée ou italique ;

— etc...

A chaque caractère imprimable comme expliqué ci-dessus correspond une configuration particulière de points. Cette configuration constitue le *graphisme* du caractère. Ces différents graphismes, dont le nombre total fournit d'ailleurs une indication précieuse sur les possibilités de l'imprimante - 64, 96, 128, voire davantage, sont stockés dans une mémoire propre à l'imprimante. Généralement de *type mémoire morte* (ROM = Read Only Memory), ces mémoires sont présentes sur des cartes logiques, dans l'imprimante. Le jeu standard de 64 caractères est généralement présent sur un ensemble de deux mémoires ROM.

Les extensions à des jeux de 96 ou 128 caractères sont rendues possibles par l'adjonction d'une ou deux mémoires ROM à la carte logique, aux emplacements prévus à cet effet. Les graphismes ainsi *stockés* sont figés ; l'utilisation de *mémoires* dites *programmables* (PROM) permet d'étendre encore la souplesse d'utilisation de telles imprimantes, en offrant à l'utilisateur toute latitude pour créer d'autres graphismes, et ce quelle que soit leur complexité.

6. CONSTITUTION STANDARD D'UNE IMPRIMANTE

Trois éléments principaux se retrouvent dans la constitution d'une imprimante.

6.1. Le bloc d'impression

Cet élément peut lui-même se décomposer en divers sous-ensembles : *le sous-ensemble tête d'impression,* lequel regroupe les éléments de frappe (aiguilles, marteaux, électro-aimants) ; *le sous-ensemble d'entraînement* de la tête (cinématique d'entraînement, moteurs continus et capteurs)(*) ; *le sous-ensemble permettant l'impression effective* - ruban encreur ou *cassette* ruban avec son propre mouvement d'entraînement.

A ces sous-ensembles fondamentaux, il y a lieu d'ajouter, pour les imprimantes issues des développements les plus récents, différentes possibilités relatives à des largeurs d'impressions différentes (les plus fréquemment répandues correspondent à l'écriture de 80 ou 132 colonnes pour une densité de 10 par pouce) ; des densités d'écritures différentes : l'impression à 10 caractères par pouce est la plus fréquente ; on notera cependant que des densités de 12 ou de 16,5 caractères au pouce sont généralement proposées. Certains types d'imprimantes disposent, enfin, en standard, de plusieurs densités d'écriture.

6.2. Le service papier

La solution à base de tracteur à chenillettes est la plus fréquemment utilisée. Selon les modèles, on pourra trouver sur l'imprimante un second tracteur à chenillettes ou un introducteur frontal de documents. Le passage du papier dépend des modèles : par *l'arrière* de l'imprimante, par *le dessous,* par *le dessus.* Le réglage de la distance tête-cylindre dépend de l'épaisseur du papier utilisé.

Quant aux mouvements de papier, leur constitution dépend des modèles d'imprimantes. Ainsi peut-on choisir entre les différentes solutions suivantes : *entraînement en continu* (le papier est alors à perforation CAROLL), deux entraînements en continu *juxtaposés,* deux entraînements en continu *avant-arrière,* et enfin, un entraînement continu et un *introducteur.*

6.3. L'électronique de commande

Celle-ci contrôle et gère le fonctionnement d'ensemble de l'imprimante ; aujourd'hui à base de microprocesseur, elle autorise, selon les modèles, l'écriture individuelle, un fonctionnement par caractère *(incrémental)* ou l'écriture bi-directionnelle, et permet de prendre en compte les *retours-têtes* (pour des ques-

(*) Ceux-ci ont par exemple pour rôle de déterminer :

— le sens de déplacement de la tête,

— la position exacte de la tête.

tions de performances globales, on essaie que ces retours se fassent à des vitesses supérieures à la vitesse d'impression), ainsi que les *tabulations rapides* (la LX 200 possède, par exemple, des vitesses de tabulation horizontale de l'ordre de 500 caractères par seconde).

7. AUTRES TYPES DE PÉRIPHÉRIQUES

L'écran/clavier et l'imprimante sont aujourd'hui les périphériques les plus utilisés pour l'entrée et la sortie des données ; c'est là la raison principale de leur description détaillée.

Ce ne sont pas cependant les seuls types de périphériques. Bien qu'ils ne soient pas (ou peu) utilisés sur les micro-ordinateurs d'aujourd'hui, nous citerons les principaux d'entre eux.

7.1. Le badge magnétique

Il permet d'introduire un petit nombre d'informations. Son utilisation est fréquente sur des postes terminaux tels que terminaux bancaire, points de vente, etc...

Les matériels de saisie de données en processus industriel ou en gestion l'utilisent également à des fins d'identification, de contrôle et de commande.

7.2. Le photostyle, ou crayon lumineux

Utilisé en liaison avec un écran, il permet de sélectionner sur celui-ci une information particulière. Très utilisé en conception assistée par ordinateur, il l'est beaucoup moins en gestion.

7.3. Le lecteur de codes à barres

Extérieurement, il ressemble beaucoup au crayon lumineux. Fonctionnellement, il permet de lire des codes se présentant sous forme de séries de barres disposées à des intervalles variables. Ces codes caractérisent l'objet sur lequel ils se trouvent : type d'article, prix, etc...

Il s'agit là d'un dispositif d'entrée de données peu onéreux ; son utilisation dans le commerce est de plus en plus fréquente. Le lecteur de codes à barres est utilisé en liaison avec un terminal point de vente.

7.4. L'entrée vocale

Basé sur l'existence d'un analyseur de voix, il s'agit là d'un périphérique d'entrée fort évolué qui, à des ordres verbaux, fait correspondre des séquences de traitement bien précises. Des machines à écrire à entrée vocale ont été annoncées

au Japon au début de 1980. L'entrée directe sur micro-ordinateur est dès aujourd'hui possible ; il s'agit là, très probablement, d'un périphérique plein d'avenir.

7.5. Sortie vocale

La restitution vocale s'appuie sur l'existence d'un synthétiseur de la voix.

7.6. Sortie graphique

C'est là une sortie écran classique, sur des tubes à haute définition. La capacité d'adresser non pas des zones d'écran mais chaque point, offre à l'utilisateur toute latitude quant au tracé de caractères, de droites, de courbes, aussi compliqués soient-ils. De tels outils sont généralement utilisés pour la conception assistée par ordinateur (études de formes, tracés divers), ou pour l'édition de courbes, d'histogrammes... (calcul scientifique, gestion financière, etc...).

7.7. Tables traçantes ou traceurs

Ce sont de véritables machines à dessiner qui, sous le contrôle de l'ordinateur et à l'aide d'informations pré-traitées, restituent des formes, des volumes ou des courbes. Les sociétés d'ingénierie, de travaux publics, les bureaux d'études de toutes les disciplines scientifiques et techniques utilisent ce type de périphérique très fréquemment (tracé d'autoroutes, ouvrages d'art, ponts, barrages, voilures d'avions, etc...).

Chapitre VII

ANALYSE D'UN SYSTÈME : LE LX 500

1. CONSTITUTION

1.1 Description générale

Le LX 500, micro-ordinateur personnel, se présente sous forme d'une boîte de dimensions modestes dans laquelle sont rassemblées :

- l'unité centrale ;
- la mémoire principale ;
- la mémoire auxiliaire de type mini-disquette, une ou deux selon les modèles ;
- l'alimentation.

Ses dimensions (largeur = 25 cm, hauteur = 16 cm et profondeur = 35 cm) le rendent particulièrement compact ; son intégration dans un poste de travail est ainsi des plus faciles, soit en le posant sur une table ou un bureau, soit tout simplement en le camouflant dans un piètement. Son poids est de 15 kg, résultant principalement d'un habillage en tôle.

La face *avant* comprend : un bouton et une ou deux fenêtres d'insertion selon que le modèle utilisé est en configuration à un ou deux lecteurs enregistreurs de mini-disquettes. La face *arrière* comprend : deux connecteurs ou voies d'entrée/sortie ainsi qu'un bouton de mise sous tension.

Le LX 500 possède deux voies d'entrée/sortie au standard CCITT Avis V24 ; cette norme, très proche du standard américain RS 232 C, définit en parti-

culier l'interface de liaison entre un *modem* (modulateur-démodulateur) et un *terminal* (le clavier-écran et le clavier-imprimante constituent les exemples les plus courants). A chaque voie d'entrée/sortie correspond un connecteur : connecteur supérieur pour la voie numéro 1, connecteur inférieur pour la voie numéro 2.

Par conception, le micro-ordinateur LX 500 joue le rôle d'un modem ; les périphériques qui, dans une configuration complète prête à l'emploi lui sont associés, jouent ainsi le rôle des terminaux.

Il est, bien entendu, possible de connecter le LX 500 à un autre ordinateur, dans ce cas, ce dernier est appelé « ordinateur maître ». La connexion se fait en intercalant entre les deux ordinateurs un dispositif appelé *bouchon* qui réalisera les croisements de fils nécessaires ; il est, en outre, nécessaire d'utiliser un programme d'initialisation adéquat.

1.2 Caractéristiques physiques et conditions d'installations

Alimentation électrique

- Tension : 110, 210, 220 ou 230 V à ± 5 %. Choix par déplacement d'un plot.
- Fréquence : 50 Hz - 60 Hz ± 10 %.
- Puissance : inférieure à 100 VA.
- Précaution à prendre : le micro-ordinateur et ses périphériques doivent être branchés sur des prises de courant avec prise de terre commune.

Température ambiante

- En fonctionnement : +4 °C à +35 °C.
- Hors fonctionnement : −40 °C à +62 °C.

Degré hygrométrique

- En fonctionnement : 20 à 80 %
- Hors fonctionnement : 1 à 95 %

Sans condensation dans les deux cas.

1.3. L'unité centrale à Z 80

Celle-ci est conçue autour d'un microprocesseur *Zilog* Z 80 qui opère à la fréquence de 4 MHz. Le Z 80 est un microprocesseur 8 bits.

L'unité centrale du microprocesseur Z 80 est capable de gérer, de comprendre et d'exécuter un ensemble de 245 instructions de base. Par combinaison des codes caractéristiques de ces instructions élémentaires, ce sont environ 700 instructions élémentaires qui peuvent être traitées.

Englobant le jeu d'instructions de son prédécesseur, le 8080 *Intel,* le langage du Z 80 offre à l'utilisateur toutes possibilités quant aux opérations de transfert et de calcul arithmétique et logique sur les données contenues en mémoire. La capacité d'adressage du Z 80 atteint un maxiumum de 64 K soit 65.536 octets.

Quelques types d'instructions figurant dans le jeu du microprocesseur Z 80 :

LOAD : Chargement d'une donnée

CP : Comparaison d'une quantité au contenu d'une quelconque position de mémoire

ADD : Addition

SUB : Soustraction

OR : OU logique

BIT : Test de la valeur d'un bit

JUMP : Saut d'une partie d'un programme à une autre partie. Un saut peut être inconditionnel ou conditionnel. Dans ce dernier cas, le saut n'est effectué que si la condition est réalisée.

1.4. La mémoire principale

De type RAM dynamique, directement liée au microprocesseur, elle offre des capacités qui peuvent évoluer de 16 à 64 K par extensions de 16 K. La valeur de 64 K correspond à la taille maximale admissible ; elle est imposée par le champ d'adressage du microprocesseur utilisé.

Les LX 500 présents dans les lycées n'ont pas de taille mémoire inférieure à 32 K, ceci pour laisser à l'utilisateur un espace mémoire de taille suffisante. En BASIC, 18 des 32 K sont disponibles, les 14 autres K étant occupés par :

— le système d'exploitation BDOS (2 K) ;
— l'interpréteur Basic (12 K).

Dans le cas d'utilisation du LSE, la mémoire centrale optimale a une capacité portée à 48 K, voire 64 K. Deux raisons conduisent à ce choix de capacité mémoire : *le souhait* de disposer d'un espace de travail utilisateur suffisant ; *la nécessité,* pour des questions de performances, de charger en mémoire la majeure partie des modules constituant le système LSE.

S'agissant de mémoire, on notera que le LX 500 dispose également d'une mémoire de type REPROM, de capacité 1 K. Celle-ci, qualifiée de mémoire morte (accessible en lecture seulement), sert à stocker les programmes de chargement initial (IPL = *Initial Program Loading* = Programme initial de chargement ou, tout simplement, programme d'initialisation) ainsi que les programmes dits de premier niveau maintenance.

En exploitation normale, c'est-à-dire dès que le processus de chargement est terminé et hors période de tests, cette zone mémoire n'est pas en conflit avec la mémoire RAM. Ainsi, l'utilisateur a-t-il bien 32 K accessibles (ou 48 ou 64, selon la configuration retenue) et non 31 (47 ou 63).

1.5. La mémoire auxiliaire

De type mini-disquette (ou mini-disque souple), elle offre une capacité qui, selon les modèles du LX 500, varie de 90 à 180 K.

Capacité 90 K : un seul lecteur/enregistreur de mini-disquette.
Capacité 180 K : deux lecteurs/enregistreurs.

Ce type de périphérique mémoire a été décrit au chapitre précédent. Rappelons simplement que ses avantages principaux ont pour noms : possibilité d'accès direct, encombrement réduit, faible coût de stockage d'un caractère, universalité, et fiabilité.

La mini-disquette constitue, en outre, un moyen simple pour diffuser programmes et données. La capacité de 90 K correspond à une utilisation du support *(média)* en simple face, simple densité.

L'évolution technologique en matière de supports magnétiques (augmentation de la densité d'enregistrement) et de lecteurs/enregistreurs, permet cependant de disposer, avec une même disquette, de capacités d'enregistrement qui peuvent être doublées, voire quadruplées. Pour cela, on met en œuvre des mini-disquettes (avec leurs dispositifs associés de lecture et d'enregistrement) qui peuvent être *double face* et/ou *double densité.*

• ***Double face :***
l'information est enregistrée sur les deux faces

capacité : $2 \times 90 = 180$ K

• ***Double face et double densité :***

capacité : $2 \times 2 \times 90 = 360$ K

Exemple :

Caractéristiques de la disquette LX 500

Taille : 5 1/4 pouces
Capacité : 90 (180) K caractères
Type : simple (double) face - simple densité
Vitesse de transfert : 15000 octets/seconde
Sectorisation : matérielle
Nombre de secteurs par piste : 10
Nombre de mots par secteur : 256
Nombre de pistes : 35
Temps d'accès moyen : 600 ou 300 ms.
Vitesse de lecture : 125 K bits par seconde.

1.6. Alimentation

Elle fournit, à partir du secteur 110 ou 220 volts, les différentes tensions nécessaires au fonctionnement du LX 500 : +5V, −5V, 12V. De type à découpage, l'alimentation du LX 500 se présente sous une forme unique, quelle que soit la tension du secteur retenu (110, 210, 220, 230 V), le passage d'une tension à l'autre se fait par simple déplacement d'un plot.

1.7. Autres caractéristiques

La face arrière est équipée, outre deux connecteurs d'entrée/sortie, d'un interrupteur de mise sous tension. Un câble, enfin, solidaire du bâti, permet de relier le LX 500 à une source d'alimentation électrique et de mise à la terre.

1.8. Architecture du LX 500

L'architecture du LX 500 est représentée dans la figure 7-1. Lors de son emploi, on prendra les précautions suivantes :

• Le LX 500 et ses périphériques doivent avoir une prise de terre commune. Les caractéristiques électriques (variations, fréquence) ayant été vérifiées, c'est là la seule précaution à prendre au plan de l'alimentation.

• Le LX 500 ne dispose d'aucun système de ventilation ; la dissipation de la chaleur dégagée par ses divers sous-systèmes le composant est facilitée par la pose de grilles d'aération sur le bâti du micro-ordinateur. *On évitera de poser quelque chose sur ces grilles d'aération.*

PRÉCAUTIONS À PRENDRE CONCERNANT LES DISQUETTES

- *Introduire les disquettes à fond mais sans forcer, en manipulant la porte de l'unité à disque avec la même main.*

- *Manipuler les disquettes en les tenant par un coin : une trace de doigt sur le support magnétique peut provoquer des erreurs.*

- *Toute disquette extraite de l'unité à disque doit être replacée dans sa pochett e, avec l'étiquette visible.*

- *Remplacer toute disquette ayant été en contact avec des liquides collants (café, sirop, ...)*

- *Attention aux aimants : ils effacent les disquettes.*

- *Attention aux objets durs, tels que ciseaux, agrafes, ... ; ils rayent les disquettes.*

- *Ne pas poser les disquettes dans un endroit chaud : radiateur, ...*

- *Stocker les disquettes verticalement avec leur pochette dans des boîtes en carton ou plastique rigide, à l'abri des poussières, de la chaleur et du froid (10 à 52 °C).*

***Nota :** On veillera à faire le plus souvent possible, suivant la nature de l'application, une duplication des disquettes contenant des fichiers, afin de pouvoir remédier le mieux possible à tout incident ayant entraîné la destruction ou la perturbation de leur contenu.*

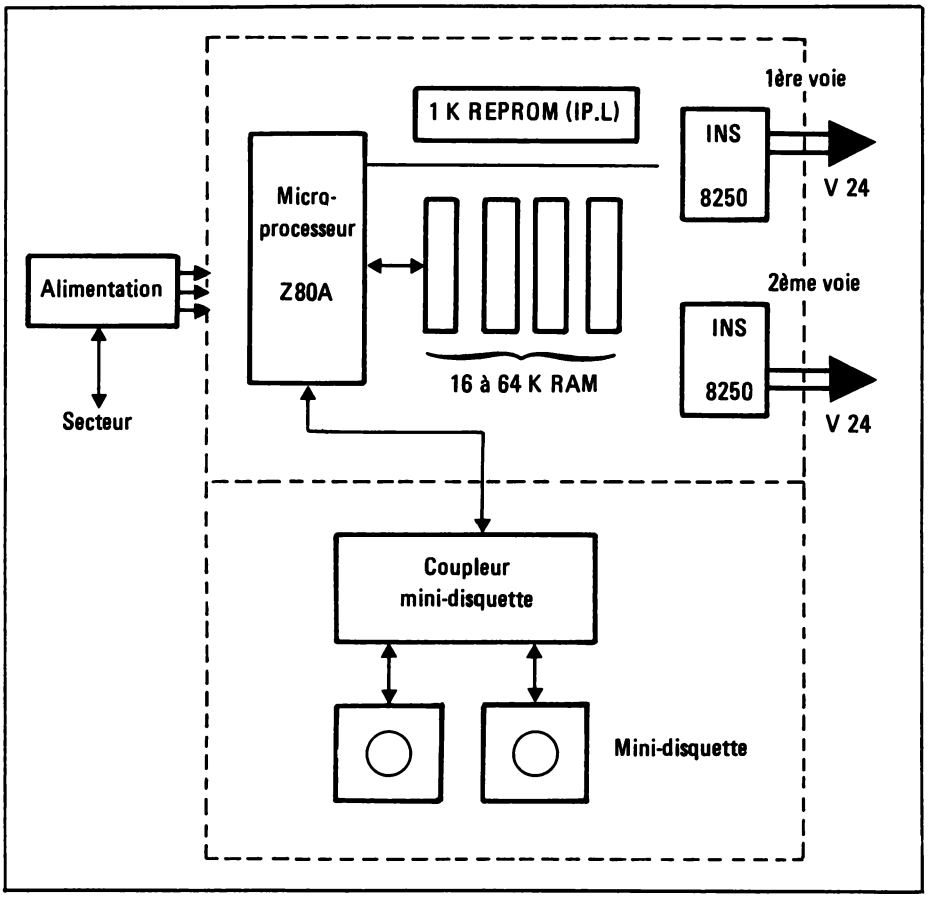


Fig. 7-1. Achitecture du micro-ordinateur LX 500.

• Lors des opérations de mise sous tension ou hors tension, vérifier qu'il n'y a pas de disquette introduite dans le (ou les) lecteur(s).

• Pour ce qui est des disquettes proprement dites, on se conformera aux précautions normalement indiquées par les fournisseurs.

2. CONFIGURATION DE TRAVAIL DU LX 500

Les deux voies « CANON » situées sur la face arrière constituent les moyens de communication du LX 500 avec son environnement extérieur. Le clavier/écran constitue le moyen le plus usuel pour communiquer :

— vers le LX 500, le clavier permet d'introduire commandes, données, instructions de programmes.

En simultanéité, l'écran permet de visualiser ces mêmes informations ;

— depuis le LX 500, affichage sur l'écran des résultats, de messages en provenance du système...

La configuration standard du LX 500 (lycées, applications de gestion, système pour dentistes, etc..) est du type LX 500 avec écran/clavier. La sortie des résultats sous forme de trace écrite, implique l'utilisation d'une imprimante. Ecran/clavier et imprimante sont reliés au LX 500 par de simples câbles. La vitesse de transmission entre le LX 500 et ses périphériques (de 110 à 19 200 bauds unité de débit équivalente, ici, à bits par seconde) est programmable. Chaque voie, ou *canal,* est programmable séparément.

2.1. Description de l'écran/clavier

Contrairement à d'autres micro-ordinateurs qui proposent l'utilisation du téléviseur grand public, comme dispositif de visualisation, le LX 500 travaille avec un écran/clavier professionnel.

D'une capacité de 2000 caractères, il permet de visualiser des informations sous forme de lignes de 80 caractères.

Le nombre maximal de lignes est ainsi de 25 se répartissant en
24 lignes + une,

la 25^{e} servant généralement à visualiser des informations en provenance du système.

Les caractères affichés apparaissent comme blancs sur fond noir ; la couleur est due au luminophore utilisé. La taille du tube, qui s'exprime par convention en fonction de la longueur de la diagonale, est d'un peu plus de 30 cm. Les caractères, d'une hauteur de 5,08 mm et d'une largeur de 2,03 mm, leur conférant une lisibilité correcte sans fatigue excessive, sont constitués par des points. Le principe de génération de caractères est ainsi du type matriciel : matrice 5×7 dans un bloc de 8×10.

A la mise sous tension, un carré blanc apparaît sur l'écran. Ce carré est appelé *curseur ;* sa fonction est d'indiquer, à tout moment, la position du prochain caractère à écrire. Afin d'être identifié dans les meilleurs délais par l'utilisateur, le curseur est animé d'un clignotement. A chaque appui sur une touche alphanumérique, le caractère correspondant apparaît sur l'écran cependant que le curseur se déplace d'une position.

En fin de ligne, le curseur se positionne automatiquement au début de la ligne suivante. S'il s'agissait de la dernière ligne, le curseur se positionnerait alors automatiquement au début de la ligne supérieure de l'écran.

Un dispositif sonore mis en œuvre automatiquement lorsqu'on atteint la place de la dixième position avant la fin de la ligne, signale à l'utilisateur qu'il est près d'avoir rempli la ligne.

Caractéristiques générales de l'écran

Ecran

— Capacité/Nombre de caractères : 2 000
— Nombre de caractères par ligne : 80
— Nombre de lignes : 25
— Ecran luminophore blanc (antireflet) : P 4 (phosphore).
— Taille du tube (diagonale) (12 inches) : 30,4 cm
— Hauteur du caractère : 5,08 mm
— Largeur du caractère : 2,03 mm
— *Méthode de balayage : trame*
— Génération de caractères : matrice 5×7 inscrits dans un bloc 8×10

Caractéristiques standard

— Entrée des données : ligne supérieure ou ligne inférieure sélectionnable
— Fin de ligne : sonnerie par interrupteur
— Nombre de caractères affichables : 126, y compris « blancs »
— Montée de ligne automatique : sélectionnable par interrupteur
— Décalage automatique d'une ligne : sélectionnable par interrupteur
— Retour automatique du curseur et saut de ligne : sélectionnable par interrupteur
— Mode moniteur : permet l'affichage des caractères de contrôle
— Positionnement du curseur : en X et Y
— Adresse du curseur : accessible en lecture et en écriture
— Contrôle du curseur : haut, bas, droite, gauche, origine, retour

Les déplacements du curseur peuvent être contrôlés par l'utilisateur lui-même. Cette faculté introduit une grande souplesse d'utilisation, l'écriture sur l'écran n'étant ainsi pas forcément séquentielle. Elle est mise en œuvre par appui sur des touches spéciales du clavier ; ces touches, identifiées par des petites flèches, permettent des déplacements du curseur dans les quatre directions :

- déplacement vers le haut ↑
- déplacement vers le bas ↓
- déplacement vers la gauche ←
- déplacement vers la droite →

Comme autres possibilités générales de l'écran, nous citerons :

• *l'inversion vidéo :* elle provoque l'écriture de caractères noirs sur fond blanc ;

• le réglage de *l'intensité lumineuse* (une intensité trop élevée peut entraîner une dégradation de l'écran).

Modes d'utilisation de l'écran

a - En local. L'appui sur une touche du clavier (c'est-à-dire la mise en œuvre du code correspondant à la touche) entraîne son affichage sur l'écran. La communication avec l'ordinateur n'est pas mise en œuvre.

b - En semi-duplex. Les codes sont envoyés simultanément sur le récepteur du terminal et sur la ligne de communication.

c - En duplex. Les codes sont uniquement envoyés sur la ligne et renvoyés en écho par l'ordinateur sur le terminal.

L'écran/clavier est normalement utilisé en tant que *terminal ;* le passage en mode *moniteur* permet d'afficher des caractères spéciaux dont les représentations graphiques sont précisées dans une mémoire PROM.

En théorie, il est possible de disposer d'un graphisme quelconque ; ceci nécessite que le différent graphisme ait été défini par l'utilisateur (sous la forme d'une grille de points) et introduit en ROM. Cette opération n'est ainsi qu'une programmation de ROM.

Caractéristiques physiques et conditions d'installation

Alimentation

— Tension : 230 V ± 10 %
— Fréquence : 50 Hz ± 2 Hz
— Puissance : 100 W nominal

Poids : 11,5 kg
Dimensions

— Largeur : 40,6 cm
— Hauteur : 34,2 cm
— Longueur : 50,8 cm

Température de fonctionnement : + 10 °C à + 40 °C
Température de stockage : − 40 °C à + 65 °C

Le clavier

Il comprend 60 touches qui permettent la génération des 128 caractères ASCII ; tous les caractères ne sont cependant pas affichables.

Le clavier standard, de type AZERTY, se compose de touches :

- ***alphanumériques** classiques :*
  - *— lettres : A à Z*
  - *— chiffres : 0 à 9*
  - *— signes : arithmétiques, ponctuation ...*
- ***de contrôle :***
  - *— exemples : retour chariot,*
    répétition,
    effacement écran,
    CNTRL
    déplacements du curseur.

En option, et physiquement séparé de l'écran, un clavier numérique réduit permet à l'utilisateur de disposer d'un moyen simple d'introduction de données exclusivement numériques.

2.2. La station d'impression

Le poste de travail élève comprend un micro-ordinateur de la famille LX 500 et un écran/clavier. Une classe peut se doter d'un nombre variable de tels postes, de 1 ou 2 à 8, 10. L'élève n'a donc pas de possibilités propres d'impression. La fonction impression est assurée, selon la définition qui en a été faite par les professeurs, par une station d'impression unique.

Constituée par une imprimante caractère, cette station est utilisée, sur demande — à la fin d'un travail par exemple — par l'élève ou le professeur. Elle est alors reliée au poste élève demandeur par l'intermédiaire d'un câble. L'imprimante utilisée est de type matriciel, une technique décrite au chapitre précédent.

3. UTILISATION

3.1. Mise en marche du LX 500

L'utilisation d'un micro-ordinateur tel le LX 500 repose sur des manipulations dont l'extrême simplicité a été voulue. Nous supposerons réalisé l'ensemble des branchements entre les divers éléments du système — unité centrale avec ses

mémoires auxiliaires, écran/clavier, imprimante, et le secteur, source d'énergie électrique. Mettre le LX 500 en état effectif d'utilisation consiste à :

— introduire dans un lecteur de disquettes, la disquette concernant le système du LX 500 ;

— appuyer sur le bouton *ON/OFF* (Marche/Arrêt).

Automatiquement, les programmes du système de base sont lus et chargés dans la mémoire centrale et le micro-ordinateur est prêt à exécuter toutes les demandes de traitement que l'utilisateur va émettre dès que le message de « FIN DE CHARGEMENT » du système apparaîtra sur l'écran.

Cette opération, qui ne demande que quelques secondes, n'est simple qu'en apparence : elle recouvre en effet une série d'opérations qui ont pour but :

• de vérifier que les éléments du système sont prêts à fonctionner ;
• que les programmes nécessaires au bon fonctionnement sont présents en mémoire.

Cette opération d'*initialisation* étant terminée, le micro-ordinateur est prêt pour utilisation ; son comportement, dorénavant, lui sera dicté par les informations qui lui seront fournies par l'utilisateur. Le type de ces informations peut être très variable et caractérise un mode d'utilisation possible.

3.2. Fonctionnement sous systèmes B DOS

L'opération de chargement terminée, le système se trouve sous le contrôle de l'utilisateur. Il peut recevoir des demandes particulières, à choisir dans un vocabulaire pré-déterminé, bien connu du système, sous la forme d'ordres. Ces ordres sont appelés *commandes.* Ils permettent, par exemple :

— de connaître le contenu de la disquette ;
— d'initialiser une nouvelle disquette ;
— de créer un fichier ;
— de copier un fichier ;
— etc...

Toutes ces opérations seront vues en détail lorsque le système d'exploitation B DOS du LX 500 sera étudié.

Nous retiendrons seulement trois notions à ce niveau :

1. L'ensemble des programmes destinés à permettre l'utilisation de la machine, et ce dans les meilleures conditions d'accès, s'appelle « Système d'exploitation », *Operating System* ou O.S., en anglais.

Dans le cas du LX 500, ce système a pour nom :

B DOS

B pour BASIC
D pour DISK ; OS pour operating system
soit : *système d'exploitation à disque, pour Basic*

2. L'accès aux fonctions permises par le système est rendu possible par l'émission de demandes, ou *commandes.*

3. A chaque commande correspond, dans le système d'exploitation, un ou plusieurs *programmes* dont l'exécution réalisera, dans les conditions fixées par l'opérateur, le traitement demandé.

3.3 Fonctionnement sous Basic

Le Basic est un langage de programmation d'utilisation relativement simple, construit autour d'un vocabulaire proche de l'anglais et disposant d'un nombre limité de mots. Les mots sont, comme dans tout langage, regroupés en phrases, et l'ensemble des phrases constitue le programme. Le LX 500 autorise deux types de fonctionnement sous Basic.

Mode Commande

Dans cet état, le LX 500 autorise la frappe :

— d'une commande ;
— d'une instruction ;
— d'une ligne de programme.

Une fois introduite, l'information est automatiquement et immédiatement analysée, décodée, et exécutée.

• *Analysée :* on vérifie qu'elle utilise des mots autorisés, connus du vocabulaire, qu'il n'y a pas de fautes d'orthographe, ...

• *Décodée :* une fois vérifiée, la phrase que constitue la commande de l'instruction ou la ligne de programme, est décomposée et interprétée. L'ordinateur lui associe un sens, détermine de quel type d'action il s'agit, et le traduit dans un langage qu'il peut exécuter. Cette phase constitue une phase d'analyse *sémantique* (sémantique : qui a trait à la signification).

• *Exécutée :* c'est la phase ultime du traitement de l'information introduite, celle qui va permettre de répondre à la demande de l'utilisateur.

L'ensemble de ces traitements fait intervenir un composant essentiel du système ; *l'interpréteur Basic.* Son avantage principal est de permettre à l'utilisateur de vérifier, immédiatement, qu'il ne s'est pas trompé en rédigeant sa demande. Si cela est le cas, il a alors la faculté de reformuler sa demande en tenant compte des renseignements fournis comme « réponse » par l'interpréteur, et ce jusqu'à ce qu'il obtienne un résultat correct.

On voit que ce mode de fonctionnement implique que l'utilisateur et l'ordinateur se posent des questions, se fournissent des réponses. Il s'agit ainsi d'un véritable *dialogue :* c'est la raison pour laquelle ce mode de fonctionnement est appelé *conversationnel.*

Mode Programme

Le mode précédent est fréquemment utilisé lors de la constitution et de la mise au point d'un programme. Lorsque le programme est prêt à être exploité, il est plus souple de le stocker de manière permanente dans une mémoire auxiliaire, de telle façon que l'on puisse le retrouver ultérieurement pour exécution.

Dans ce cas, le mode de fonctionnement s'intitule « mode programme ». Lu sur la mini-disquette, mémoire auxiliaire du LX 500, chargé en mémoire centrale pour exécution, le programme commence à s'exécuter à la réception d'une commande spéciale de début d'exécution (RUN), et celle-ci se poursuit, sauf incidents dus, par exemple, à une mise au point insuffisante ou sauf intervention particulière et voulue de l'utilisateur, jusqu'à l'instruction de fin de programme (STOP).

Ces quelques explications sur le mode de fonctionnement du LX 500 nous ont ainsi amené à introduire des notions fort diverses.
— système d'exploitation ;
— langage et interpréteur BASIC ;
— commandes d'initialisation ou de copie de disquettes, de fichiers, ...

Nous avons vu également que nous pouvions communiquer de deux façons avec l'ordinateur.
• au moyen de commandes ;
• au moyen de programmes.

Ces notions sont propres à tous les ordinateurs et non pas uniquement aux micro-ordinateurs du type LX 500 ; selon le degré de complexité de l'ordinateur, on trouvera bien entendu des possibilités plus grandes au plan du dialogue homme-machine, tant pour ce qui concerne le dialogue proprement dit (langages de programmation nombreux, langage de contrôle, etc.) que pour la manière d'utiliser les supports mémoires auxiliaires (bandes, disques magnétiques), et les divers dispositifs périphériques (écrans/claviers, lecteurs de cartes, de badges, imprimantes...)

Chapitre VIII

DIALOGUE ET PROGRAMMATION

Au premier abord, il y a deux manières d'utiliser votre ordinateur :

• Ou bien le programme existe *(et il est alors enregistré dans une mémoire auxiliaire, une disquette par exemple) et il faut seulement connaître le langage de commande pour « l'appeler » et le faire « exécuter » avec vos données.*

• Ou bien le travail souhaité n'est pas encore « programmé » *(ou du moins, le programme n'existe pas dans votre bibliothèque de disquettes) et il faut alors apprendre le langage de commande... mais aussi un langage de programmation :*

BASIC

ou

LSE...

Dans ce chapitre, nous allons construire progressivement un petit programme, étudier les principales « instructions » du langage BASIC afin d'apprendre à nous servir de l'ordinateur pour résoudre les exercices proposés par ailleurs, ou, tout simplement, commencer à programmer de petites applications.

Mais attention, *ce ne sont pas ces rudiments qui feront de vous un programmeur, car programmer, c'est avant tout analyser le problème, le décomposer en « structures » et « algorithmes » connus, sujets qui sortent du cadre de cet ouvrage.*

Les notions traitées ici seront complétées dans le chapitre suivant, puis dans celui consacré plus spécialement et plus complètement à la programmation en Basic. A titre d'exercices, on trouvera ensuite quelques programmes complets.

Nous vous conseillons de lire ce chapitre face au micro-ordinateur et d'exécuter la suite de tous les exercices qui sont proposés.

Votre micro-ordinateur LX 500 est-il branché ? La console est-elle connectée ? La lumière rouge allumée ?

On peut alors introduire une disquette fournie avec le système par le fabricant et comportant des programmes de service dont on examinera ce qu'ils sont plus loin. On verra apparaître, dans le cas du LX 500, « BDOS-1-S » à l'écran (ou d'autres informations avec d'autres micro-ordinateurs : il conviendra de se référer, à chaque fois, à la notice d'emploi proposée avec la machine ; ceci est valable pour la suite de cet exposé).

En cas d'ennui :
— appuyer sur le bouton RESET* ;
— enlever puis introduire à nouveau la disquette ;
— au pire, prendre une autre disquette dont vous savez qu'elle contient le système BDOS et le Basic ;
— le BDOS de la disquette est-il conforme aux vitesses de transmission du terminal (300 bauds ou 9 600 bauds par exemple) ?

Dès que * BDOS *... apparaît, tapez B ; Basic suivi de RETURN ou RC... (suivant la console).

Après le petit bruit dû à la lecture sur la disquette, le message « BASIC... » doit apparaître à l'écran, suivi du mot « COMMAND ».

Vous êtes alors « sous Basic » et l'interpréteur Basic attend vos commandes.

* Le seul bouton de la façade sous le voyant rouge, sur le LX 500. Pour les autres machines, se référer au manuel d'emploi.

1. L'UTILISATION ACTIVE

Nous avons choisi de vous faire programmer un petit exemple de facturation pour... un livreur de fuel.

A chaque client livré, le livreur doit indiquer le montant de son dû. Soit à facturer 1 300 litres de fuel à 2,20 F le litre. Deux modes sont possibles, examinons-les :

Mode bureau

Montant dû : $2{,}20 \times 1\,300$

Mode Basic

Interviennent ici les contraintes du langage Basic :
1) — Le signe « multiplié » s'écrit « * » (pour ne pas le confondre avec la lettre « x »).
2) — On n'écrit pas les nombres décimaux avec une *virgule,* mais avec un *point* (ceci est dû au fait que les premiers langages de programmation furent créés aux Etats-Unis et que les Américains utilisent le point comme séparateur décimal).

Donc : **2.20 * 1300**

Si vous tapez cela, l'ordinateur fera peut-être le calcul mais n'affichera pas le résultat *tant qu'on ne le lui demande pas.*

L'instruction d'affichage est :

PRINT

(qui veut dire *imprimer,* en anglais, et correspond au fait que les premiers ordinateurs ne « sortaient » les résultats que sur une imprimante ou sur une machine à écrire) ;

```
PRINT  2.20 * 1300  (RC)*
2860
```

Résultat obtenu, notre ordinateur est une bonne calculette...

* Nous utiliserons les lettres RC entre parenthèses pour dire « retour du chariot » ou « return »... ; tout dépend de la console qui vous fait face.

Mais dans une calculette, il faut à chaque fois redonner toutes les valeurs numériques alors que l'ordinateur dispose d'une mémoire. Il suffit de donner un NOM (un *identificateur*) à chaque valeur que l'on veut manipuler. En Basic, ce nom doit être une lettre (une seule), ou une lettre suivie d'un chiffre.

Exercice :

Combien de noms différents peut-on utiliser en Basic ?

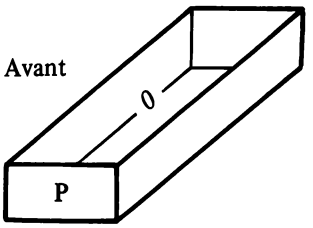
Par exemple, le prix d'un litre de fuel sera nommé P (« P » comme... Prix). Encore faut-il l'initialiser (lui donner une valeur) :

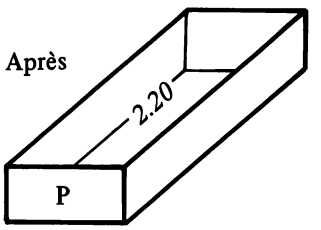
```
P = 2.20
```

Ceci s'appelle une « instruction d'affectation ». Donc :

```
P = 2.20  (RC)
PRINT P * 1300  (RC)
2860
PRINT P * 400
880
(etc.)
```

RC est la commande de « retour chariot », rappelons-le.





Comment effacer un caractère que l'on vient de frapper ?
Tout simplement avec la touche d'EFFACEMENT (qui peut s'appeler DEL ou RUB ou BACK-SPACE...) Pour permettre de vérifier que la commande a bien été exécutée :
- *soit le caractère disparaît vraiment de l'écran ;*
- *soit, au contraire, il s'affiche à nouveau, en « écho » comme on dit.*

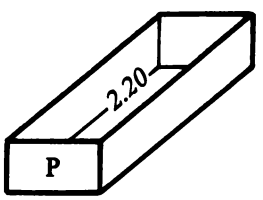
Essayez... au pire, vous retaperez la ligne.

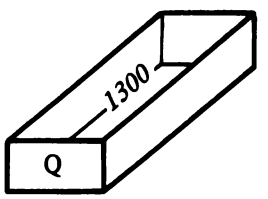
PRINT est un peu long à taper, nous dirons désormais PR.
Mais attention, c'est la SEULE abréviation autorisée en Basic.

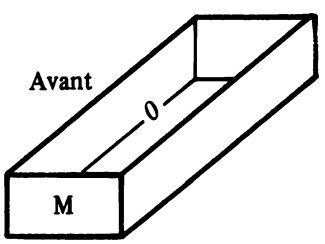
```
PRINT P * 100
220
PR P * 100
220
```

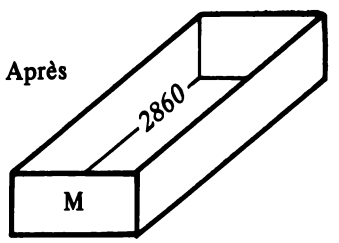
Généralisons : en Basic, les valeurs manipulées, (les variables) reçoivent un nom. Le programme décrit les opérations à effectuer en utilisant ces identificateurs. Notre exemple devient alors :

```
P=2.20
Q=1300
M=P * Q
PR M
2860
```









Toutes ces instructions sont exécutées au fur et à mesure (cela s'appellera le MODE BUREAU) mais elles ne sont pas mémorisées et à chaque nouveau client, le livreur devra redemander :

```
Q= ...
PR P * Q
.....
```

Nous allons maintenant créer un véritable programme et le mémoriser. En Basic, il suffit pour cela de numéroter les instructions ; alors, Basic les enregistre dans l'ordre des numéros (que nous appellerons des ETIQUETTES) :

```
1    P=2.20
2    Q=1300
3    M=P * Q
4    PR M
```

Tout cela s'enregistre dans la mémoire mais ne s'exécute pas, contrairement à tout à l'heure.

1.1 Comment faire exécuter ?

RUN	**La commande RUN lance l'exécution du programme et la première instruction exécutée est celle qui a la plus petite étiquette, même si les instructions ont été rentrées dans le désordre.**

Essayons. Mais pour cela, il nous faut tout effacer en mémoire grâce à la commande CLEAR (à ne pas confondre avec la touche CLEAR, qui existe sur certaines consoles et dont le seul rôle est d'effacer l'écran, mais ne touche en rien au programme en mémoire). Allons-y :

```
CLEAR    (RC)
LIST     (RC)
```

Cette nouvelle commande nous permet de faire « lister » le programme contenu en mémoire... Bien entendu, après un CLEAR, la mémoire est vierge de toute instruction

1.2 Numérotation de 10 en 10

Comme il est possible d'oublier des instructions (mais il vaut mieux bien réfléchir avant de programmer car cela ne devrait pas arriver), ou encore de vouloir ajouter des modifications, on préférera souvent numéroter de 10 en 10 :

```
CLEAR
40      PR M
10      P=2.20
30      M=P * Q
20      Q=1300

RUN

2860

COMMAND
LIST
10      P=2.20
20      Q=1300
30      M=P * Q
40      PR M
20      Q=100

RUN

220

COMMAND
LIST

...
```

En cas d'erreur, on retape toute la ligne, avec le même numéro et la ligne précédente sera effacée

Mais 2860 s'affiche seul à l'écran et ce n'est guère explicite. Aussi va-t-on améliorer l'instruction 40.

```
40      PRINT " MONTANT DU  :     ";M
```

Tout ce qui se trouve entre les deux apostrophes sera affiché sur l'écran avant la valeur de M :

```
RUN
MONTANT DU  :     2860
```

Allons plus loin :

```
40    PRINT " MONTANT DU  :    ";M;"  FRANCS "

RUN
MONTANT DU  :    2860 FRANCS
```

Petit exercice :

Essayer de faire afficher

..... × ..... = ... *FRANCS*

Le PRIX unitaire ***La QUANTITE commandée*** ***Le MONTANT dû***

1.3. L'INPUT

Ce programme calcule le montant dû pour Q = 1300. Mais pour un autre client ?.. Doit-on faire un autre programme ? Non, il suffit d'avoir une instruction qui, à l'exécution, va venir chercher une valeur au clavier et l'affecter à une variable. Ici, par exemple, nous remplacerons

```
20    Q = 1300
```

par

```
20    INPUT Q
```

Essayons :

```
20    INPUT Q
LIST
.....
.....
RUN
?
```

Ce point d'interrogation signifie que l'ordinateur est dans l'attente d'une donnée. Introduisons-la.

```
? 1300 (RC)
MONTANT DU  :     2860 FRANCS
COMMAND

RUN
? 200
MONTANT DU  :     440 FRANCS
COMMAND
```

> ***Comment effacer une ligne ?***
>
> *On tape son numéro (sans blanc derrière) suivi d'un (RC) qui commande, rappelons-le, un " retour de chariot ".*

Seulement, « ? » tout seul à l'écran n'est pas très explicite et dans des programmes plus complexes, on risque fort de ne pas se souvenir de ce qu'il faut donner. On utilise alors un « libellé » entre deux apostrophes qui viendra s'afficher à l'écran à la place du point d'interrogation :

```
20    INPUT " QUELLE EST LA QUANTITÉ  ?  :  ";Q
```

1.4. La boucle

Actuellement, il faut demander un RUN à chaque client et le programme s'exécute en entier à chaque fois ; pour l'éviter, il suffirait de demander à revenir à l'instruction 20 après chaque client :

```
50    GOTO 20
```

> *GOTO (*) et non GO TO*
> *Les mots comme PRINT, INPUT, GOTO sont appelés des mots-clés et il ne faut pas de blancs à l'intérieur.*

Inlassablement, après chaque résultat, le programme redemande une quantité. Ce programme " boucle ", comme on dit. Pour s'en sortir :

1) *Couper le courant.....* alors on perd tout. (N'oublions pas que la mémoire centrale de l'ordinateur est « volatile » et que tout arrêt d'alimentation électrique en fait perdre le contenu.)

2) Il est plus sérieux d'utiliser un arrêt prévu par Basic : tout dépend des consoles, mais en général, on utilise deux touches : « CTRL » que l'on doit maintenir appuyée et « C ». Alors, on « sort » du programme et Basic demande une nouvelle commande.

> *TRÈS IMPORTANT*
> *Retirez les disquettes avant d'éteindre l'ordinateur !*

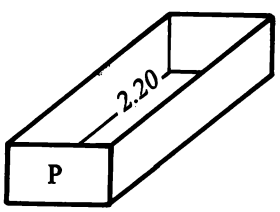
(*) En anglais, GOTO = aller en, allez à !

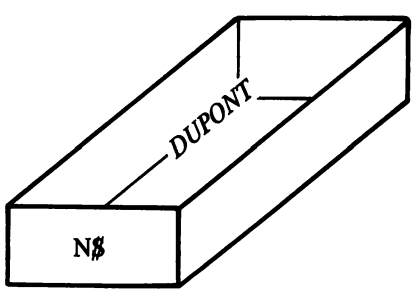
1.5. Du nom du client

Nous avons manipulé des nombres (désignés par *un identificateur* : une lettre, éventuellement suivie d'un chiffre). Mais on peut aussi manipuler des noms propres, des adresses, etc., par exemple le nom du client.

Ces *variables* (qu'on appelle « variables alphanumériques * », ou encore « chaînes de caractères »), doivent être distinguées des variables numériques car aussi bien la manière dont elles sont mémorisées que la façon de les traiter sont très différentes.

L'identificateur d'une chaîne de caractère comporte, en Basic, le symbole " $ " après une lettre éventuellement suivi d'un chiffre.





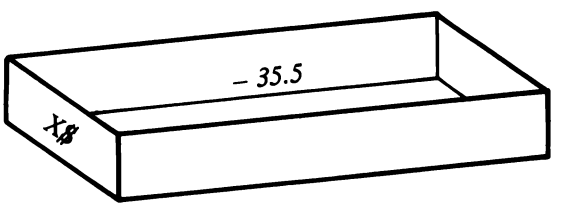
- La variable P identifie un nombre.
- La variable N$, ... une chaîne de caractères, ici : le nom du client.

(*) Suite de lettres ou de chiffres, ou même de tout autre caractère du clavier (et, comme nous verrons plus tard, tout autre caractère de commande du clavier et/ou de l'écran ; par exemple, le code de la petite sonnerie « tûûût »).

Attention

• *On peut rencontrer des CHIFFRES dans une suite de caractères :*

• *Un nombre lui-même peut être considéré comme une suite de chiffres et (éventuellement) de symboles : " – " ou " + " pour le signe et " . " pour la virgule ..... Mais on ne pourra pas faire d'opérations arithmétiques avec :*



Que peut-on faire avec ces variables alphanumériques et comment les manipuler ?

Tout d'abord, on peut affecter une valeur à une variable en utilisant le même symbole = que pour les variables numériques, mais en plaçant la valeur alphanumérique entre apostrophes.

Exemple : N$ = " DUPONT "
X$ = " – 35.5 "

Une remarque très importante : les " blancs ", les intervalles, créés par la barre d'espacement du clavier sont très importants... Surtout quand ils ne sont pas visibles à l'écran, en début ou en fin de mot :

" DUPONT " est différent de " DUPONT "

...et ils comptent comme des caractères alphanumériques.

Essayons d'entrer le petit bout de programme suivant :

```
100     N$ = " DUPONT "
```

Si vous avez la réponse " type error ", c'est que vous êtes toujours en train d'exécuter votre programme et que Basic attend un nombre et non pas de l'alphanumérique... Alors, faites " CTRL " et " C " pour retrouver l'interpréteur Basic.

```
110     PRINT N$
RUN     100
```

Pourquoi RUN 100 ?

Effectivement, nous voulons seulement exécuter ces deux instructions et non pas tout le programme.

La commande RUN lance l'exécution de l'instruction qui a la plus petite étiquette, RUN 100 lance l'exécution à partir de l'étiquette 100 :

```
RUN 100
DUPONT
COMMAND
?
```

Essayons à nouveau avec un nom plus long :

```
100     N$ = " DUPONT DE LA DURANDIÈRE "
RUN 100
DUPONT DE
.....
```

Sans autre indication, les " boîtes " alphanumériques sont ouvertes à 10 caractères alphanumériques (les blancs comptent).

La taille (ou la dimension) d'une variable alphanumérique dont la longueur dépasse 10 (ou dépassera 10) doit être demandée, avant tout emploi de cette variable, par l'instruction DIM qui spécifie après la variable et entre parenthèses la longueur maximale de la chaîne de caractère N$:

```
90 DIM N$ (23)
RUN 90
DUPONT DE LA DURANDIÈRE
.....
```

Là encore, le nom du client va changer, d'un client à l'autre et le mieux est de faire un " INPUT " du nom du client :

Rappelez ici le listage du programme en cours par la commande LIST.

Il semble normal de demander le nom du client avant la quantité désirée ; c'est donc avant 20, mais après 10 que l'on doit faire cette entrée, donc en 15 :

```
15 INPUT N$
```

où mieux :

```
15 INPUT " NOM DU CLIENT S.V.P. ? " ; N$
```

Essayez d'exécuter avec plusieurs noms et n'oubliez pas de sortir de la boucle par CTRL, C.

Vous remarquez que l'on revient à la " quantité commandée ? " (instruction 20) et non à la 15. Il faut changer :

```
50 GOTO 20   en
50 GOTO 15
```

puis dimensionner le nom du client à...20 voire 30 caractères avant l'instruction 15 :

```
5 DIM N$ (30)
```

Reprenons :

```
LIST
.....
```

Du fait que 90, 100 et 110 sont inutiles ; supprimez les !

```
90              (RC)
100             (RC)
110             (RC)
LIST            (RC)
____
____
____
RUN             (RC)
```

Nous avons entré le nom du client .... autant le sortir dans la facture et commencer à faire de la mise en page.

Essayons de faire afficher :

MR. XXXXXXXXX
DOIT YYYYY FRANCS

avec une ligne blanche avant et une autre après.

En Basic, on affiche une ligne blanche à l'écran avec l'instruction :

```
n     PRINT (RC)
```

(PRINT suivi du RETURN)

Pour le reste, vous devez savoir écrire le programme, à l'adresse 38 par exemple :

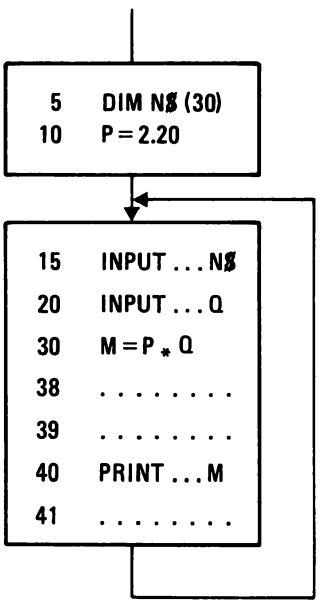
```
38    PRINT
39    PRINT " MR.  ou  ME.  "; N$
40    PRINT " DOIT  " ; M ; "  FRANCS. "
41    PRINT
```

Faites LIST et RUN, pour voir ce qui se passera.

2. D'UNE REPRÉSENTATION GRAPHIQUE DU PROGRAMME

Plus un programme s'allonge et se complique, plus il devient difficile de " voir " sa structure, son organisation en regardant le listage. Aussi peut-on utiliser une représentation graphique pour le représenter.

Ici, par exemple, la structure de boucle apparaît nettement si l'on remplace le GOTO par une flèche de retour et que les instructions sont regroupées en pavés :



Pour obtenir une représentation plus propre :

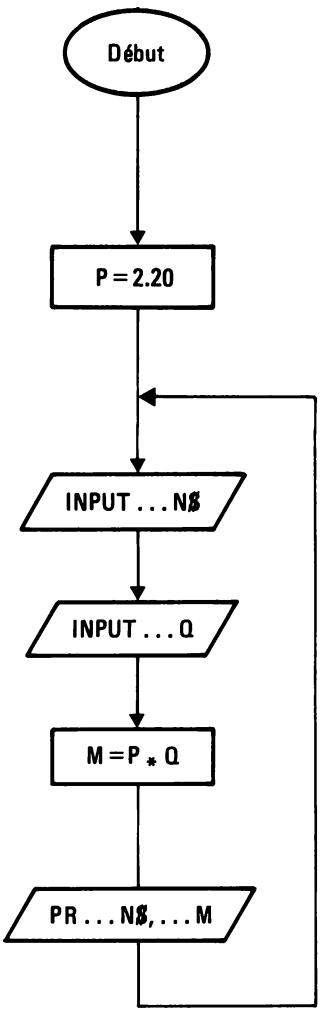
— On ajoute une bulle (Début) pour marquer le point de départ de l'exécution.

— On encadre les instructions d'affectation et, plus généralement, les instructions de traitement, dans des rectangles.

— On encadre les instructions d'entrée de données et de sortie des résultats dans des parallélogrammes.

— On supprime les instructions de déclaration (comme DIM) et les instructions d'organisation (comme GOTO).

— On peut même supprimer les étiquettes puisque les flèches décrivent la succession des opérations à effectuer :



Ce dessin s'appelle un *organigramme* et ce type de représentation pourra servir à beaucoup d'autres processus.

3. LE TEST

Tous les clients subissent le même traitement et c'est la même succession d'instructions qui est exécutée pour chacun d'eux.

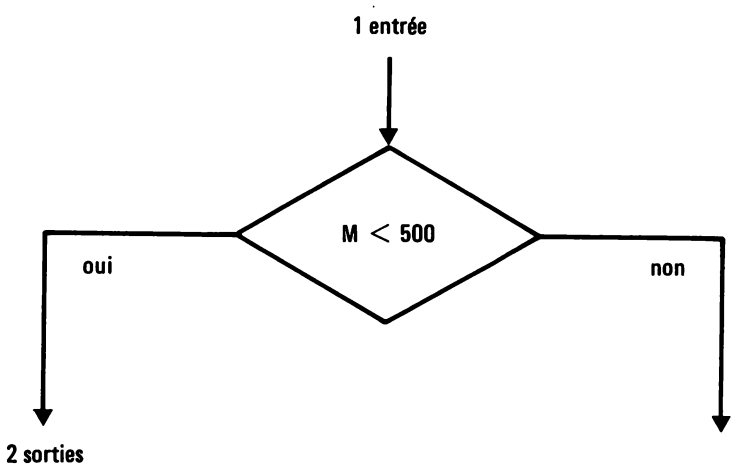
Imaginons qu'il y ait une majoration de frais de transport lorsque le montant de la livraison est trop faible. Soit à rajouter 50 F à toute livraison dont le montant est inférieur à ...500 F.

A l'issu du calcul du montant dû, M, il faut introduire un test : *est-ce que M est inférieur à 500 ?*

- *Si oui, le montant est majoré de 50 ;*
- *Si non, le montant est inchangé.*

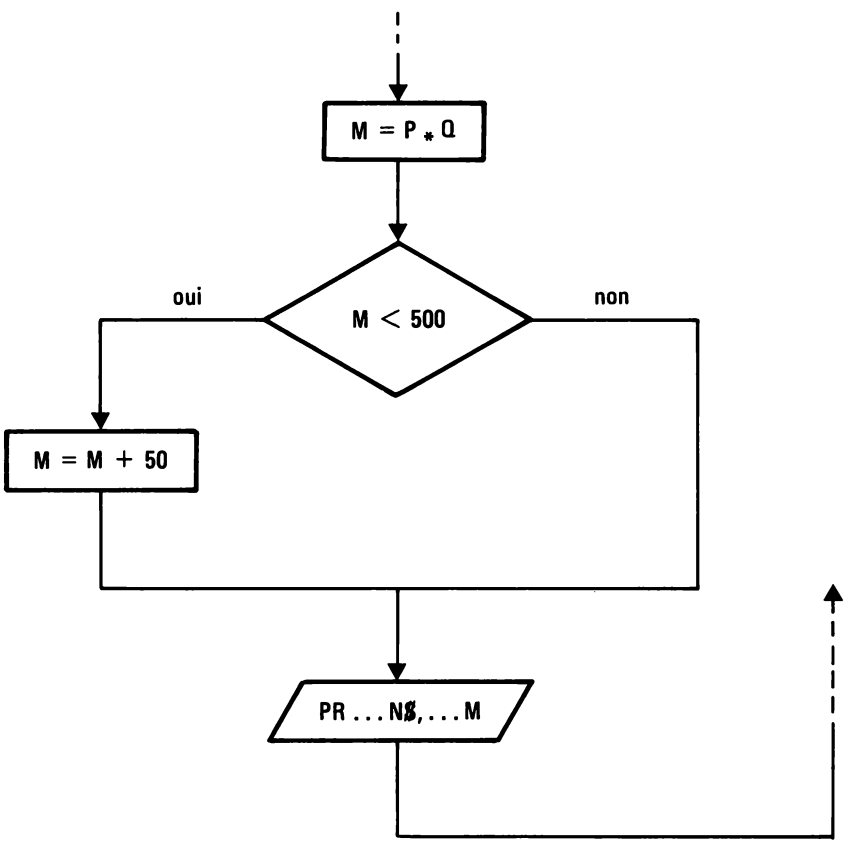
> *Remarquons que l'ordinateur travaille sur des nombres indépendamment de leur signification. On n'écrira donc pas 500 F ou 50 F, mais 500 ou 50, seul un libellé " FRANCS " dans l'ordre PRINT fera figurer cette signification.*

Sur l'organigramme, un test s'introduit à l'aide d'un losange :

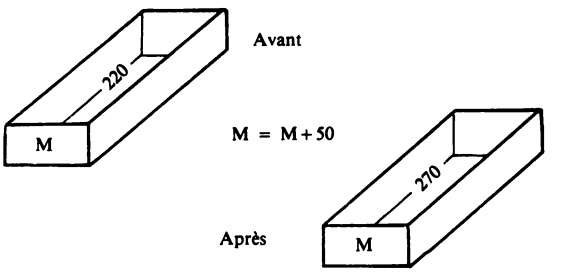


Le losange renferme une question dont la réponse ne doit être que " oui " ou " non " et, suivant cette réponse, la suite à exécuter sera placée sur l'une ou l'autre branche.

Ici par exemple



L'écriture M = M+50 a de quoi choquer au premier abord, mais n'oublions pas que M représente le « contenu » d'une zone mémoire.
M = M+50 signifie donc « Prendre la valeur de M, lui ajouter 50 et ranger le résultat (la nouvelle valeur de M) dans la même zone mémoire » :



Pour supprimer cette ambiguïté du signe =, plusieurs langages de programmation ont codé l'affectation par une flèche :
M←M+50

3.1 L'INSTRUCTION IF

En Basic, l'alternative est réalisée par l'instruction :

IF ... THEN ... ELSE ...
(si) ... (alors) ... (sinon) ...

Donc :

IF (une condition) THEN *(une instruction)* ELSE *(une instruction)*

« Si » la condition est réalisée, « alors » l'instruction suivante sera celle qui suit le THEN, « sinon », ce sera celle qui suit le ELSE.

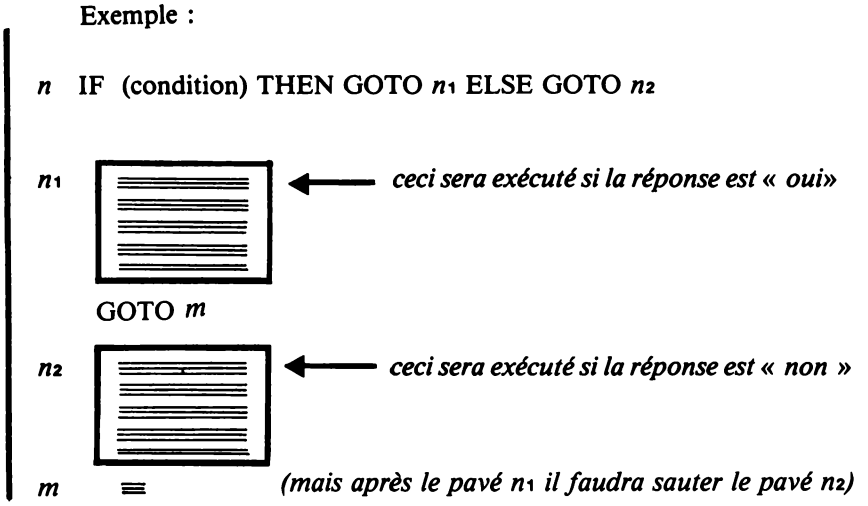
Pour notre exemple, on peut écrire

```
35    IF M< 500 THEN M = M + 50 ELSE M = M
```

On constate que pour la réponse « non » au test, il n'y a rien à effectuer : la syntaxe du IF accepte que l'on supprime le ELSE lorsqu'il est inutile, l'instruction suivante à effectuer étant celle qui suit le programme.

Remarquons tout de suite qu'on peut utiliser l'instruction GOTO *n* à la suite du THEN, ou du ELSE, et ainsi on pourra, suivant le résultat d'un test, se brancher à l'un ou à l'autre des deux autres « pavés » d'instructions.

Exemple :


n IF (condition) THEN GOTO n1 ELSE GOTO n2
n1
ceci sera exécuté si la réponse est « oui»
GOTO m
n2
ceci sera exécuté si la réponse est « non »
m
(mais après le pavé n1 il faudra sauter le pavé n2)

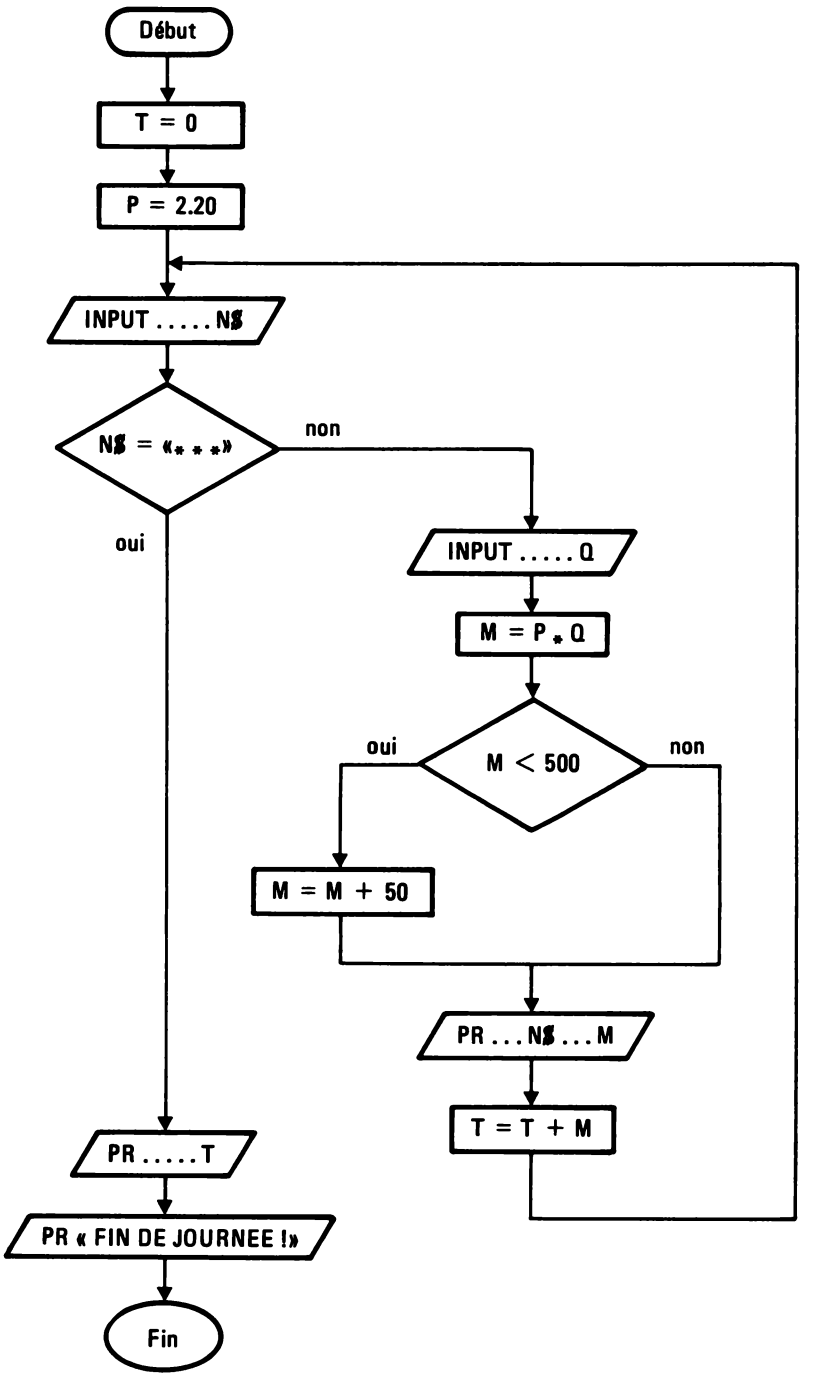
3.2 Test de fin de boucle

Pour « sortir » d'une boucle, nous n'avons pas encore d'autre moyen que de stopper l'exécution par « CTRL » et « C ». Or, on peut avoir à exécuter un boucle un certain nombre de fois, puis à reprendre une autre séquence d'instructions. Il est donc devenu classique de terminer l'exécution d'une boucle et de se brancher à une autre séquence d'instructions en utilisant un test.

Notre dernier client aura un nom particulier qui signifiera que nous désirons « sortir de la boucle ». Ainsi, à la demande du nom du dernier client, nous taperons quelque chose qui ne doit, en aucune façon, être un nom de vrai client et par exemple « *** ».

L'instruction IF nous permet de tester le nom du client et de sortir de la boucle !

Visualisons la chose :



Et rédigeons la séquence :

```
17    IF N$ = " *** " THEN GOTO 80
80    PRINT " FIN DE JOURNEE ! "
90    STOP
```

Faites un LIST puis un RUN.

Après avoir traité plusieurs clients, donnez *** comme nom de client ; vous devez terminer votre programme lorsque les instructions 80 et 90 seront exécutées.

Mais avant le STOP, vous auriez pu créer tout un programme en dehors de la 1re boucle.

4. LA NUMEROTATION DES INSTRUCTIONS

Le programme qu'on a examiné dans le chapitre précédent a maintenant des étiquettes quelque peu disparates. Basic autorise la renumérotation de 10 en 10 par la commande :

```
NUMB (RC)
LIST (RC)
10    DIM N$ (30)
20    P = 2.20
30    INPUT " NOM DU CLIENT S.V.P. ? " ; N$
40    IF N$ = " *** " THEN 130
50    INPUT " QUELLE EST LA QUANTITE ? : " ; Q
60    M = P*Q
70    IF M < 500 THEN M = M + 50
80    PRINT
90    PRINT " MR. OU ME. " ; N$
100   PRINT " DOIT  " ; M ; "FRANCS. "
110   PRINT
120   GOTO 30
130   PRINT " FIN DE JOURNEE ! "
140   STOP
```

On remarque que les étiquettes sont changées « en suivant » dans les IF et les GOTO « GOTO 15 » est devenu « GOTO 30 ».

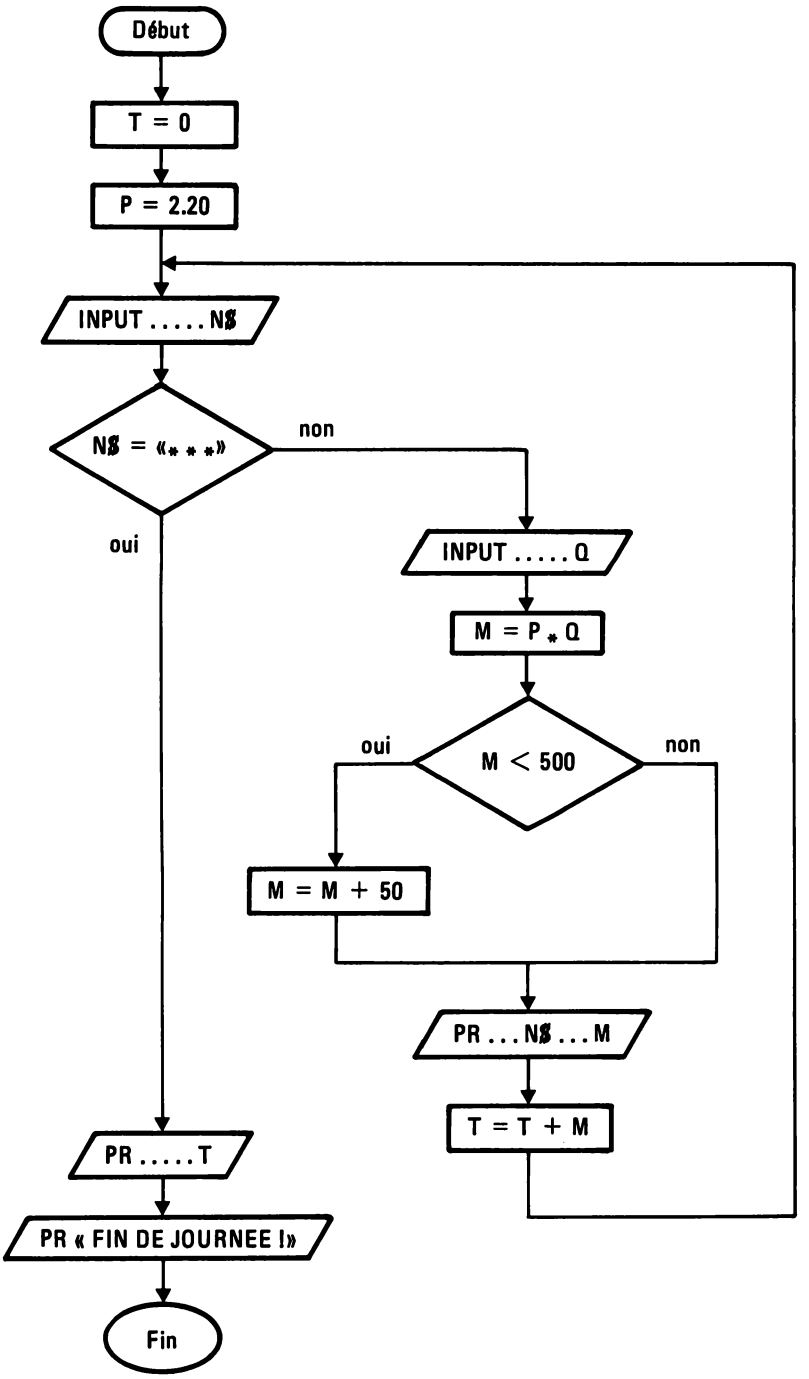
5. NOTION DE SOMMATION

Pour terminer ce programme de livraison, nous allons effectuer la totalisation du montant des factures du jour.

On pourrait, bien entendu, demander au livreur de noter les montants successifs, de vérifier les niveaux de cuve au départ et à l'arrivée..., mais il est plus commode de créer un TOTALISATEUR dans lequel nous mettrons 0 en début de programme, puis où nous cumulerons les montants successifs.

Dans l'organigramme, T = 0 doit intervenir dès le début, puis à chaque facturation, nous cumulerons les montants M dans T et ferons T = T + M, d'où l'organigramme général et le programme en Basic (après NUMB) :

```
10    DIM M$ (30)
20    T = 0
30    P = 2.20
40    INPUT " NOM DU CLIENT S.V.P. ? " ; N$
50    IF N$ = " *** " THEN 150
60    INPUT " QUELLE EST LA QUANTITE ? : " ; Q
70    M = P*Q
80    IF M < 500  THEN  M = M + 50
90    PRINT
100   PRINT " MR. OU ME. " ; N$
110   PRINT " DOIT   " ; M ; "FRANCS. "
120   PRINT
130   T = T + M
140   GOTO 40
150   PRINT " TOTAL DU JOUR " ; T
160   PRINT " FIN DE JOURNEE ! "
170   STOP
```


Début
T = 0
P = 2.20
INPUT . . . . . N$
N$ = «* * *»
non
oui
INPUT . . . . . Q
M = P * Q
oui
M < 500
non
M = M + 50
PR . . . N$ . . . M
T = T + M
PR . . . . . T
PR « FIN DE JOURNEE !»
Fin

Etes-vous équipé d'une imprimante ? Si oui, branchez-la sur la voie 1 du LX 500 (prise du haut) et commandez tout simplement :

```
LIST  @1; (RC)
```

Si tout va bien, vous obtiendrez un listage de votre programme sur papier.

Chapitre IX

LE SYSTÈME D'EXPLOITATION

1. LES TROIS PROGRAMMES

Le fonctionnement de l'ordinateur est commandé par trois programmes.

- le BDOS *(Basic Disk Operating System)* pour gérer les disquettes ;
- le Basic pour manipuler des programmes ;
- le programme de l'utilisateur pour traiter ses données.

On peut passer de l'un à l'autre par des commandes ou des instructions spécifiques à chacun qui sont résumées dans la liste suivante (fig. 9-1).

1.1. Principales commandes du BDOS

L pour lister le contenu de la disquette.

C ; FACTUR ; 5 ; P pour créer une zone de 5 secteurs qui contiendra un programme nommé FACTUR.

S ; FACTUR pour supprimer la zone FACTUR (et donc perdre son contenu).

B ; BASIC pour charger le BASIC en mémoire.

B pour revenir au BASIC après l'avoir quitté sous QUIT.

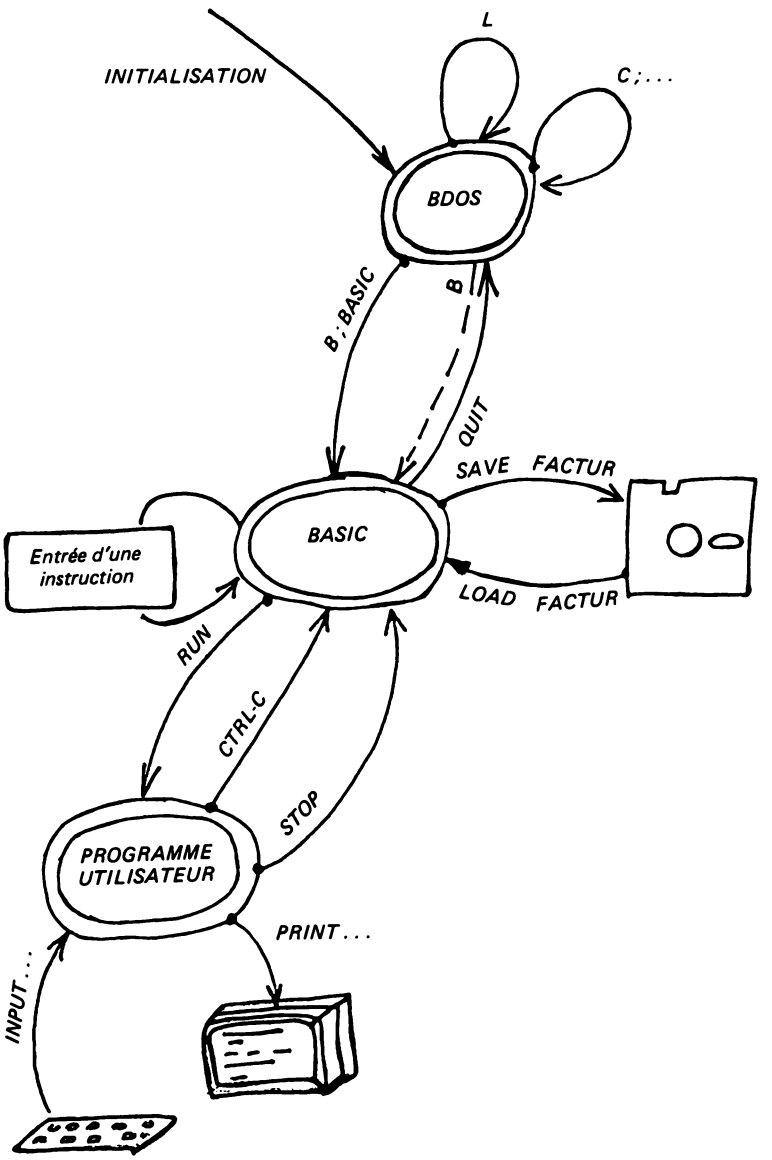


Fig. 9.1 — Trois types de programmes peuvent intervenir.

1.2. Principales commandes du Basic

LIST pour lister le programme en cours contenu dans la mémoire centrale.

CLEAR pour effacer le programme de la mémoire centrale (aucun effet sur la disquette).

SAVE FACTUR pour sauvegarder le programme sur la disquette sous le nom FACTUR.

LOAD FACTUR pour charger le programme FACTUR, de la disquette vers la mémoire (en écrasant le programme précédent).

RUN pour lancer l'exécution du programme en cours.

QUIT pour revenir au BDOS.

1.3 Commande du progamme utilisateur

STOP pour arrêter le programme et retourner au Basic.

1.4 Commandes extérieures

L'initialisation (chargement du BDOS) et CTRL-C (arrêt d'un programme en exécution et retour au Basic) sont, en fait, deux « commandes » extérieures qui interviennent lorsqu'on appuie sur le bouton-poussoir de la façade, ou sur la touche « contrôle » du clavier, mais qui ne sont ni des commandes Basic ou BDOS, ni des instructions de programme.

2. SAUVEGARDE D'UN PROGRAMME ET UTILISATION DU BDOS

Votre programme est actuellement enregistré dans la mémoire centrale de l'ordinateur, mais si vous coupez l'alimentation électrique ou si vous voulez entrer un nouveau programme, vous allez le détruire. Comment mémoriser un programme sur une disquette (fig. 9-2) ?

Les disquettes sont gérées par le BDOS *(Basic Disk Operating System)* qui tient à jour une sorte de tableau ou de répertoire dans lequel on trouve :

— le nom des fichiers contenus sur la disquette ;

— leur adresse sur la disquette ;

— leur longueur (en nombre de secteurs) ;

— leur type :
- fichier binaire (ce sont BDOS et BASIC (le traducteur)) ;
- fichier de programme (comme FACTUR que nous voulons mémoriser) ;
- fichier de données :

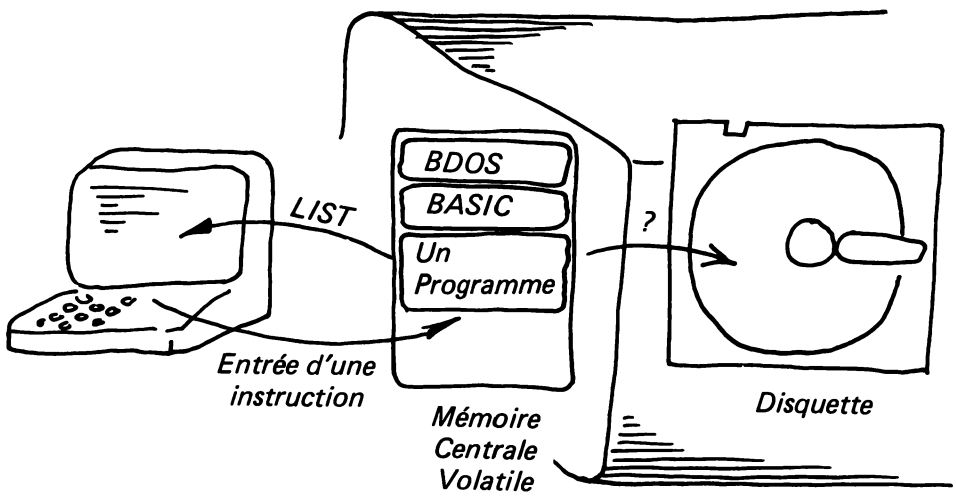


Fig. 9.2 — Utilisation d'une disquette.

L'utilisation du BDOS permet de créer de nouveaux fichiers dans la mesure de la place disponible sur la disquette. Il faut alors :

1) Quitter l'interpréteur BASIC et revenir sous BDOS :

2) Utiliser BDOS pour créer un fichier programme en lui donnant le nom que vous voulez attribuer à votre programme ;

3) Revenir sous BASIC;

4) Sauvegarder (ou garer) votre programme sur la disquette.

En BASIC cette manipulation ne supporte aucune erreur qui risque de détruire le programme en mémoire centrale, alors *ATTENTION !*

COMMAND
? QUIT *(RC)*

Cette commande nous fait " **passer** " *sous contrôle du BDOS.*

? C ; FACTUR ; 5 ; P (RC)

C'est un fichier " **PROGRAMME BASIC** " : **P pour programme**
B pour binaire
D pour données

Donne la taille réservée en secteurs ; ici, nous réservons une zone de **5 × 256 = 1280** *caractères pour mémoriser le programme.*

Notre programme sera stocké sous ce nom et nous permettra de le retrouver plus tard sur la disquette.

C *pour* " **CREER** ".

? B *(RC)*

Pour revenir sous Basic sans détruire le programme en cours. **ATTENTION, LA FRAPPE DE B ; BASIC REMET LA MEMOIRE A VIDE ET DETRUIT LE PROGRAMME EN COURS.**

COMMAND
? LIST

Pour vérifier la présence du programme.

S.....
.....
.....
COMMAND
? SAVE FACTUR

Sauvegarde le programme présent en mémoire centrale, sur la zone disquette que nous avons créée sous le nom **FACTUR.**

XXX bytes

donne la taille exacte **(mesurée en caractères)** *du programme au moment de sa sauvegarde.*

3. EXPLOITATION D'UN PROGRAMME ET UTILISATION DU BASIC

Comment utiliser des programmes déjà enregistrés sur une disquette (soit qu'elle vous ait été fournie, soit que vous l'ayez créée vous-même antérieurement) ?

Il faut, tout d'abord, placer l'ordinateur sous contrôle de l'interpréteur Basic et pour cela, comme précédemment :

— brancher ordinateur et console ;
— les alimenter en courant électrique ;
— introduire une disquette contenant BDOS et BASIC (avec les vitesses de transmission adaptées à votre terminal) dans le lecteur de gauche ;
— fermer la porte du lecteur de disquette et attendre l'apparition du libellé :

* BDOS .....

si un petit rectangle apparaît, il faut appuyer à nouveau sur le bouton de façade.

Avec le libellé * BDOS, le système d'exploitation offre les différentes possibilités de commande résumées au paragraphe 1.1

L *(RC)*

pour lister le contenu de la disquette :

C ; FACTUR ; 5 ; P *(RC)*

pour créer une zone fichier qui comprendra un programme de moins de 5 secteurs de volume (moins de 1280 caractères).

B ; BASIC (RC)

pour appeler l'interpréteur Basic et pouvoir, soit créer un nouveau programme, soit utiliser un programme déjà enregistré.

Dès que

```
COMMAND.
?
```

apparaît, vous êtes sous Basic et vous pouvez utiliser les commandes du Basic, rappelées au paragraphe 1.2.

```
LOAD FACTUR (RC)
```

pour " charger ". *Après une lecture disquette et quelques secondes, le programme FACTUR réside en mémoire centrale.*

```
LIST (RC)
```

donne le listage complet du programme et nous replace dans les conditions du chapitre précédent.

Maintenant, vous pouvez lancer une exécution de ce programme par

```
RUN (RC)
```

ou le modifier en introduisant de nouvelles instructions, et par exemple :

```
5 PRINT " * EXECUTION D'UNE FACTURATION * "
6 PRINT
```

qui fera apparaître ce titre suivi d'une ligne blanche en tête de l'exécution.

Mais on peut aussi changer une instruction, par exemple, en remplaçant l'affectation P = 2.20 par une entrée de donnée :

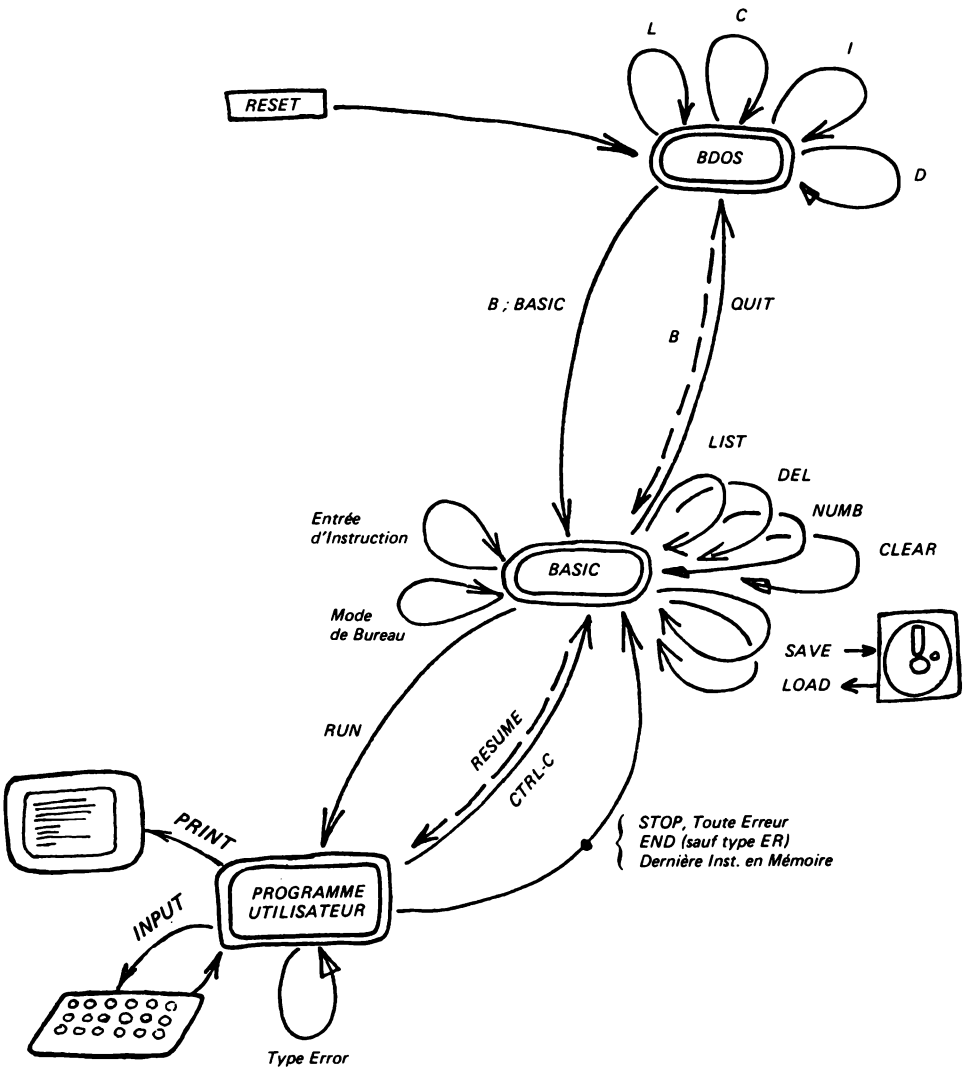


Fig. 9.3 — Petite marguerite pour s'y retrouver.

```
30 INPUT " QUEL EST LE PRIX UNITAIRE ? : " ; P
```

A chaque exécution de ce nouveau programme, on pourra faire un nouveau prix unitaire.

> ***Exercice :***
>
> *Modifiez de même le programme pour que la valeur limite du « Franco de port » (ici 500 F) et le montant de la taxe soient variables à chaque exécution du programme.*

Bien entendu, la disquette ne mémorise que l'ancienne version du programme ; si vous désirez conserver le nouvelle version modifiée, n'oubliez pas de :

```
SAVE FACTUR (RC)
```

Ce faisant, vous détruisez l'ancienne version. Aussi est-il bon ,... en phase de mise au point, de conserver une version 1, une 2,... sous différents noms.

A vous de créer les fichiers programmes nécessaires.

Chapitre X

LES PROGRAMMES ET LEUR EXÉCUTION

1. MODES D'UTILISATION D'UN MICRO-ORDINATEUR

1.1. La monoprogrammation

Dans un tel contexte, l'ensemble des moyens offerts par l'ordinateur est à disposition d'un seul l'utilisateur.

Quels sont ces moyens ? Rappelons-les :

— l'unité centrale ;
— la mémoire centrale et la (ou les) mémoire(s) auxiliaire(s) ;
— les dispositifs d'entrée/sortie tels l'écran/clavier, l'imprimante ;
— les programmes, qu'il s'agisse des programmes systèmes (le BDOS et le BASIC ou le LSE dans le cas du LX 500), ou du programme d'application proprement dit.

Tous ces moyens sont indispensables à l'utilisateur pour réaliser le traitement désiré ; l'absence d'un seul d'entre eux rend impossible le traitement. On dit que tous ces éléments constituent *les ressources* ordinateur. Un seul utilisateur travaillant à un instant donné, il n'y a aucun conflit d'accès aux différentes ressources. Ce mode de fonctionnement, le plus simple que nous verrons, est appelé *monoprogrammation ;* il caractérise tous les micro-ordinateurs présents aujourd'hui sur le marché, dont le LX 500.

Accédant à ces différentes ressources, soit par commandes, soit par programme, l'utilisateur indique à l'ordinateur quel type de traitement utiliser. Chaque type de traitement constitue un *travail,* chaque travail comprenant une ou

plusieurs *étapes*. Une même étape peut présenter une ou plusieurs phases de type traitement ou de type entrée/sortie.

Exemple :

On souhaite lire deux quantités A et B, les additionner, mettre le résultat de l'addition dans une variable dénommée C et imprimer le résultat.

Nous avons ainsi, comme opérations à réaliser :

• *Deux lectures*	*Lire A* *Lire B*
• *Une addition avec affectation du résultat à C*	*A + B* *C = A + B*
• *Une écriture*	*Écrire C*

Le fonctionnement du micro-ordinateur est alors le suivant :

.... *Entrée A*	*Entrée B*	*Traitement*	*Sortie C* ....
Lire *A*	Lire *B*	*C = A + B*	Écrire *C*
phase (n – 2)	(n – 1)	(n)	(n + 1)

Programme

En résumé, dans la monoprogrammation :

• l'ordinateur travaille pour un seul utilisateur à la fois ;
• il n'y a qu'un seul programme dans tout le système, à quelque niveau que ce soit : unité centrale ou mémoire centrale ;
• pour réaliser l'opération *n*, la machine doit avoir terminé de réaliser l'opération *n* – 1 ;
• une seule zone mémoire sert aux diverses opérations d'entrée, de traitement ou de sortie d'information.

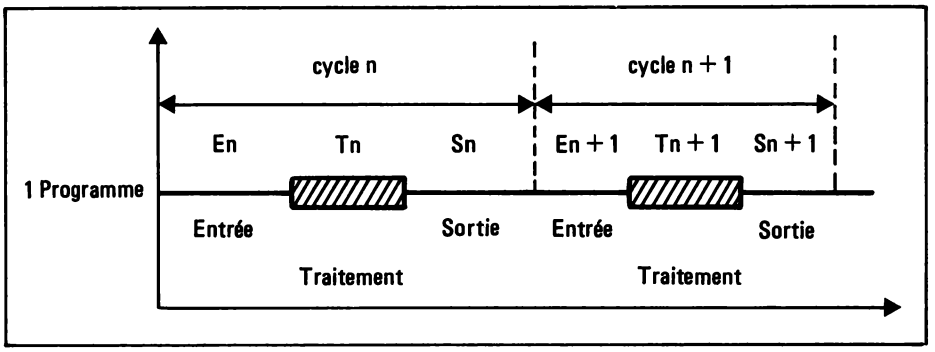


Fig. 10.1 — Déroulement d'une tâche en monoprogrammation et en mode direct.

Mode direct et mode indirect

Ce mode est qualifié de mode direct (fig. 10-1) ; il existe, en effet, un autre mode dit « indirect » qui caractérise une exploitation séquentielle de travaux. Le mode indirect n'est cependant pas utilisé dans les micro-ordinateurs mais plutôt sur des ordinateurs de puissance (en termes de vitesse de traitement et de diversité des configurations) supérieure. Dans ce cas, plusieurs programmes utilisateurs co-existent dans le système. On notera cependant qu'à un instant donné, un seul programme est en exécution. Il s'agit ainsi de monoprogrammation. Les autres programmes sont en attente, et le passage d'un travail au travail suivant ne se fera que lorsque le travail en cours sera terminé. Le passage de l'un à l'autre se fait sous contrôle d'un programme particulier du système, dénommé « *moniteur d'enchaînement* ».

Simultanéité des entrées/sorties

Les opérations de traitement proprement dites et les opérations d'entrée/sortie se font de manière séquentielle. Or, l'étude des principaux composants de l'ordinateur et celle des différents types de périphériques ont montré que les vitesses de traitement étaient fort différentes, les vitesses au niveau des périphériques étant très sensiblement inférieures.

Il en résulte que les capacités de l'unité centrale ne sont pas intégralement utilisées, d'où une baisse de la performance globale.

Afin de pallier cet inconvénient, certains systèmes font appel à la possibilité de chevauchement dans le temps des opérations d'entrée/sortie avec les opérations de traitement proprement dites.

On dit qu'il y a *simultanéité :* cette propriété est mise en œuvre par un ensemble spécifique de programme dénommé IOCS - *Input/Output Control System* - ou *système de contrôle des entrées/sorties.* Le schéma de la figure 10-2 illustre cette propriété de simultanéité.

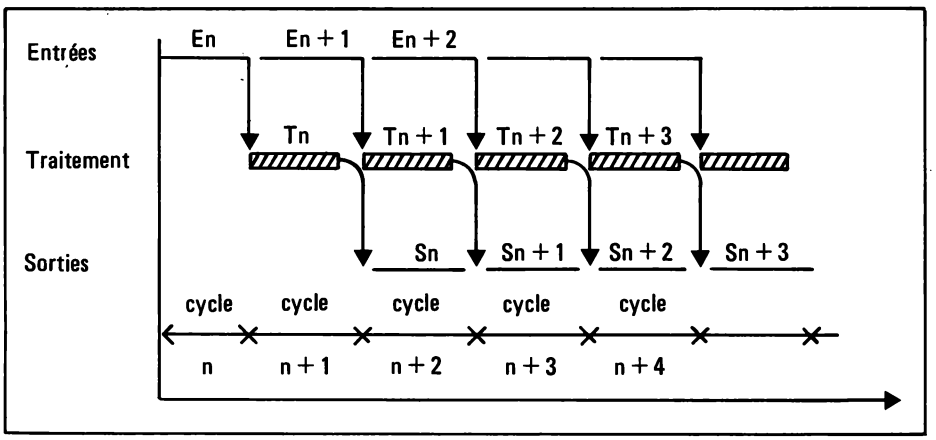


Fig. 10.2 — Ici, les entrées-sorties s'effectuent en simultanéité avec le traitement.

Bien que simplifié, ce schéma montre aisément que la performance globale de l'ordinateur est multipliée par près de 3 (un peu moins, en réalité, car les entrées/sorties sont notablement plus longues que les traitements).

Afin d'éviter des pertes d'informations, plusieurs zones mémoires sont utilisées pour les entrées et les sorties. La gestion de ces zones est assurée par les programmes constituant l'IOCS.

IOCS

Mis en œuvre à chaque instruction d'entrée/sortie.

Fonctions

- *Détermination de la zone mémoire à utiliser.*
- *Mise en liaison avec le périphérique concerné.*
- *Contrôle du bon déroulement du transfert de l'information.*
- *Branchement à la partie de programme devant se dérouler simultanément à l'entrée/sortie.*

D'autres modes d'exploitation des ordinateurs sont proposés. Ils ne concernent pas les matériels généralement groupés dans la famille des micro-ordinateurs ; aussi seront-ils seulement énumérés ici pour mémoire.

1.2. La multi-programmation

On dit qu'un système est exploité en mode *multi-programmation* lorsque plusieurs programmes utilisateurs résident simultanément pour exécution, en mémoire centrale, et que leur exécution se fait de manière imbriquée. L'imbrication fait appel à un *système d'interruptions,* notion que nous introduirons ci-après. Il existe deux types de multi-programmation.

Multi-programmation sans interruption

Lorsque, durant l'exécution d'un programme, l'on aboutit à une entrée/sortie, le *programme de contrôle* du système (ou *moniteur*) lui retire le contrôle au profit d'un autre programme qui, de ce fait, se voit attribuer la ressource unité centrale (fig. 10-3).

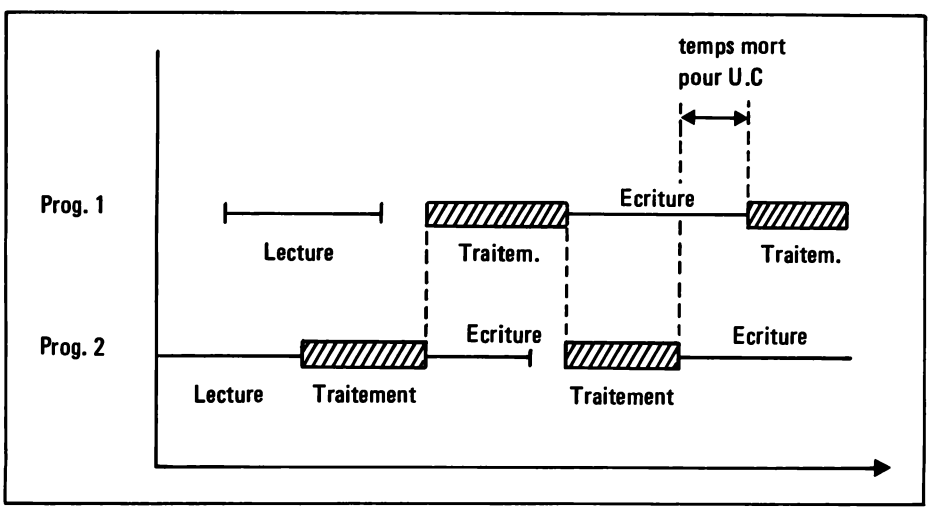


Fig. 10.3 — Multiprogrammation, appliquée à deux programmes : les traitements interviennent pour l'un quand l'autre passe à des entrées-sorties.

Multi-programmation avec interruption de programme

Dans ce cas, les programmes simultanément résidents se voient affecter une priorité. On dit que les programmes sont *hiérarchisés.* Supposons que deux programmes P1 et P2 soient résidents en mémoire. Affectons à P2 la priorité la plus

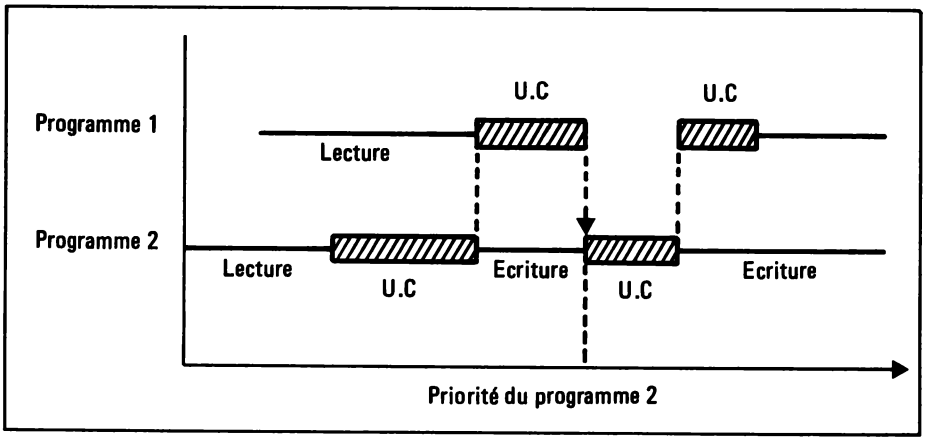


Fig. 10.4 — Multi-programmation avec interruption : chaque programme prévient l'autre dès que son traitement est terminé.

élevée, et à P1 une priorité moindre. Dès que le programme P2 aura terminé son opération d'écriture, le périphérique le signale au programme de contrôle en transmettant un code spécial (fig. 10-4). Cet envoi constitue une *interruption de programme ;* cette interruption étant prioritaire (P2), le programme P1 est interrompu au profit du programme P2 qui peut ainsi poursuivre son exécution.

1.3. Le temps partagé

Cette dénomination caractérise un mode d'exploitation de l'ordinateur dans lequel la ressource « temps » est successivement mise à disposition de l'utilisateur. Une tranche de temps *(quantum)* d'unité centrale est allouée à un utilisateur ; lorsque celle-ci est épuisée, écoulée, une autre tranche de temps est allouée à un autre utilisateur.

On s'assure ainsi d'une meilleure gestion de la ressource unité centrale, sans pour autant pénaliser chaque utilisateur. Celui-ci a, en effet, durant la tranche de temps qui lui est affectée, l'accès à toutes les ressources de l'ordinateur. Autorisant la présence simultanée de plusieurs utilisateurs sur une même configuration, ce mode d'exploitation se caractérise ainsi par un rapport performance/prix extrêmement avantageux.

1.4. Le multi-traitement

Le multi-traitement est généralement défini comme assurant la coexistence, dans *une même machine,* de *plusieurs unités centrales.* Celles-ci travaillent simultanément sur des programmes ou des morceaux de programmes différents. Le multi-traitement est associé à la notion de *multiprocesseur.*

Tout d'abord introduite sur de très grosses machines, cette notion est aujourd'hui présente dans de nombreux systèmes de traitement (systèmes de gestion où coexistent processeur arithmétique et processeur fichier, terminaux dits intelligents présentant des capacités autonomes de prétraitement avec un processeur écran et un processeur pour les échanges avec le monde extérieur, etc.). Elle n'est pas encore très répandue dans les micro-ordinateurs ; rien, en théorie, ne s'y oppose.

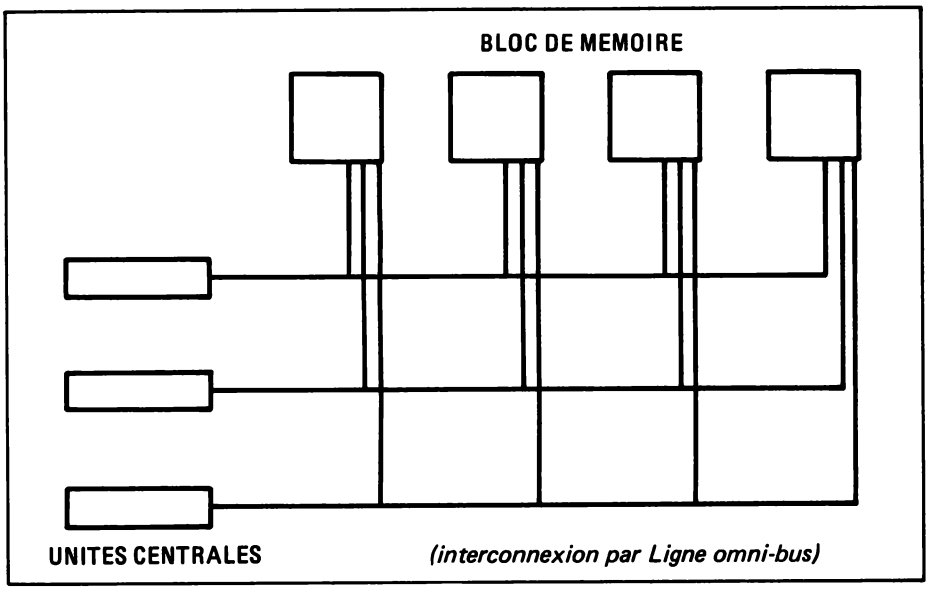


Fig. 10.5 — En multi-traitement, on dispose de plusieurs unités centrales qui se partagent, si nécessaire, plusieurs blocs de mémoire.

L'utilisation d'un multi-processeur débouche tout naturellement sur les notions de mémoire partagée et de mémoire dédiée (fig. 10-5).

Mémoire partagée

Le multi-processeur a accès à une seule mémoire qui constitue la mémoire centrale de l'ordinateur. Chaque processeur ne peut accéder qu'à une portion de cette mémoire ; celle-ci est dite *partagée.* Des dispositifs de protection interdisent à un processeur d'accéder à toute la mémoire.

Mémoire dédiée

A chaque processeur est associée une mémoire principale. Les communications avec les autres mémoires sont physiquement impossibles.

Intérêt du multi-traitement

• La multiplication des éléments pouvant travailler en simultanéité permet d'accroître la puissance du système.

• La présence de plusieurs processeurs permet, lorsqu'ils sont banalisés, d'accroître la fiabilité du système.

Cette propriété est en particulier utilisée dans les systèmes de contrôle de lancement et de guidage des fusées, de pilotage de certains processus industriels (coulée d'un haut fourneau, contrôle d'une centrale...).

1.5. Le temps réel

L'information est traitée dès qu'elle prend naissance au niveau de la périphérie et le résultat retransmis immédiatement à cette même périphérie. L'exploitation en temps réel se caractérise ainsi par une *prise en compte immédiate ;* de ce fait, le temps de réponse, c'est-à-dire le temps qui s'écoule entre un événement et l'action devant résulter de l'occurrence de cet événement, est extrêmement bref. C'est cette brièveté du temps de réponse qui a fait donner à ce type d'exploitation la dénomination d'exploitation en temps réel.

Une exploitation en temps réel est généralement requise dans tout type de traitement où le temps a une influence notable.

Exemples :

1. Contrôle de la température de l'huile dans un moteur :

Des capteurs de température mesurent à tout instant la température à des endroits bien déterminés d'un moteur. Si l'un d'eux mesure une température anormalement élevée, par exemple, il envoie un signal qui constitue le point de départ d'une séquence d'alarme pouvant éventuellement déclencher l'arrêt du moteur.

Ne pas traiter l'information en temps réel pourrait conduire à une valeur de la température trop forte avec des risques tels que explosion, incendie, etc.

2. Guidage d'une fusée :

A chaque instant, les paramètres de vol d'une fusée (vitesse, altitude, azimut, inclinaison, etc.) sont transmis à la station terrestre de contrôle. Toute valeur anormale de l'un des paramètres peut conduire la fusée à un fonctionnement non prévu. En temps réel, l'ordinateur envoie des corrections destinées à la remettre sur la trajectoire correcte. Si le fonctionnement est par trop anormal, l'ordinateur peut envoyer un ordre de destruction pour éviter que la fusée ne provoque, par exemple, des dégâts sur des zones habitées.

Dans ces deux cas, la valeur du temps de réponse doit être extrêmement faible. N'oublions pas, par exemple, qu'une fusée peut parcourir des kilomètres en une seconde.

3. Autres cas

Il est d'autres cas où le temps de réponse peut être de l'ordre de quelques secondes, voire de quelques minutes. La réservation de places (avion, chemin de fer), la consultation d'un compte bancaire, le retrait d'espèces sur un compte chèque postal ou sur un compte bancaire, constituent quelques exemples d'ores et déjà courants.

2. SYSTEME D'EXPLOITATION

L'étude du déroulement du travail d'un utilisateur et des divers modes d'exploitation des ordinateurs nous ont permis d'introduire des notions relatives aux :

— programmes de contrôle-moniteur ;
— traducteurs de langages, compilateurs, interpréteurs, assembleurs ;
— systèmes de contrôle des entrées/sorties ;
— etc.

L'ensemble de ces notions constitue un tout homogène, caractéristique d'un ordinateur, que l'on appelle « *système d'exploitation* ». Un système d'exploitation comprend généralement deux grandes familles de programmes.

2.1. Programmes privilégiés

Ce sont des programmes conçus par le constructeur même de l'ordinateur, que l'utilisateur n'a pas à modifier. Ils regroupent les programmes de gestion des :

— commandes ;
— travaux ;
— programmes ;
— données ;

et ont pour noms :

— superviseur, moniteur d'enchaînement ;
— contrôleur d'entrées/sorties ;
— interpréteur de commandes ;
— allocateur de mémoire ;
— ...

2.2. Programmes non privilégiés

On regroupe généralement sous cette rubrique des programmes écrits :

• *par le constructeur :*

— processeurs de langages (BASIC, LSE, ...) ;
— programmes de service ou utilitaires (initialisation de fichiers, copie de fichiers, ...) ;

• *par l'utilisateur lui-même :*

— il s'agit de programmes d'application.

2.3. Notion de traducteur de langage

De par sa constitution, l'ordinateur ne sait reconnaître que des ensembles reposant sur l'utilisation de deux états qui, par convention, sont représentés par 0 et 1. L'utilisation du système binaire n'est cependant pas des plus simples ainsi qu'on l'a vu précédemment.

Aussi a-t-on introduit des langages d'utilisation plus proches dans leur syntaxe, des langages naturels. Ces langages, dits langages de programmation, sont de types fort divers, chaque type s'appliquant à une ou plusieurs familles d'applications.

Des exemples de tels langages sont fournis par :

• Le FORTRAN (contraction de *formula translation* = traduction de formules).

Ce langage est principalement destiné aux programmes d'applications scientifiques ou techniques.

• Le COBOL (contraction de *Common Business Oriented Language,* ou langage commun orienté vers des applications administratives et commerciales), pour la gestion.

FORTRAN et COBOL ont été définis et mis en application il y a près d'une vingtaine d'années. Depuis lors, bien d'autres langages, plus universels, c'est-à-dire moins orientés dans leurs applications, ont vu le jour : PL 1 (Programme Langage n° 1), APL *(A Programming Language),* BASIC enfin (Beginner's All Purpose Symbolic Instruction Code), en sont des exemples typiques.

Tous présentent comme caractéristique d'utiliser des vocabulaires fort proches de l'anglais.

> ***Exemples :***
> *DO... est un verbe d'action du FORTRAN qui indique à l'ordinateur de FAIRE ce qui est précisé après.*
>
> *IF... est également un mot clé du langage FORTRAN qui, à la condition précisée à sa suite, indique à l'ordinateur quelles séquences d'instructions il y a lieu d'exécuter.*
>
> *FOR... NEXT est, en BASIC, l'équivalent du DO... CONTINUE du langage FORTRAN.*

Cette simplicité, des langages de programmation n'est cependant qu'apparente ; toute instruction de quelque langage que ce soit doit en effet être analysée,

décodée, interprétée, modifiée en une suite de 0 et de 1, seules informations assimilables par l'ordinateur. Ces opérations n'étant plus faites par l'utilisateur lui-même, un organe doit les réaliser. Cet organe a pour nom : *traducteur.*

Le traducteur

La traduction, en langage absolu, assimilable par l'ordinateur, d'un programme écrit en langage symbolique, fait intervenir un programme spécialement conçu à cet effet, le traducteur. *Traducteur, compilateur, processeur,* sont alors des termes équivalents.

2.4. Système de programmation

L'ensemble du langage symbolique et de son compilateur constitue le système de programmation.

Exemples :

- *Système de programmation BASIC = ensemble des règles d'écriture des instructions BASIC plus le traducteur associé, interpréteur ou compilateur.*

- *De même, le système de programmation FORTRAN est constitué par l'ensemble langage FORTRAN-traducteur FORTRAN.*

Si les règles définissant un langage peuvent être extrêmement générales, il n'en est pas de même pour les traducteurs. Ceux-ci en effet, par définition, conduisent à des chaînes binaires de 0 et de 1, dont les configurations sont typiques de l'ordinateur utilisé. Les traducteurs sont ainsi propres aux machines sur lesquelles ils sont installées.

Un système de programmation peut être utilisable sur plusieurs ordinateurs : ces ordinateurs sont dits *compatibles* entre eux. Par contre, le langage absolu est caractéristique de l'ordinateur.

— Un langage commun est un langage de programmation symbolique conçu pour être utilisé sur différents ordinateurs.

— Le langage absolu est un langage de programmation proche des caractéristiques internes de la machine, conçu par les réalisateurs mêmes de la construction de l'ordinateur.

En résumé, *pour un ordinateur précis,* il y a :
— un ou plusieurs langages symboliques ;
— un seul langage absolu.

Exemple :

• Le LX 500 a un seul langage absolu ; il s'agit de l'assembleur ZILOG, du nom du microprocesseur utilisé.

• Le LX 500 peut être utilisé avec plusieurs langages symboliques ; BASIC et LSE sont les exemples les plus courants ; APL, FORTRAN sont d'autres exemples. A chaque langage correspond un traducteur.

• Le système de programmation LX 500 comprend ainsi en réalité plusieurs sous-systèmes de programmation : BASIC, LSE, FORTRAN, etc.

Chapitre XI

PROGRAMMONS EN BASIC

Les auteurs veulent être très clairs sur l'apprentissage de la programmation présenté ici. Rappelons que pour programmer il faut :

— connaître le langage de programmation (mots du langage, forme des instructions, instructions d'organisation du programme, ...),

— savoir « analyser » un problème, c'est-à-dire, extraire les structures des données, formaliser les traitements adéquats, etc., toutes choses qu'un seul chapitre ne peut ni exposer ni même montrer.

C'est pourquoi ce chapitre se limitera à présenter les instructions les plus élémentaires du langage BASIC mises en œuvre dans plusieurs petits programmes.

Nous vous conseillons, à chaque exercice, d'en essayer l'étude et la programmation BASIC, dans plusieurs cas, nous vous proposerons des extensions sans en donner la solution qui reste à votre portée compte tenu des notions présentées et des exemples traités.

Dès que l'ordinateur est « sous BASIC » (et alors le mot COMMAND : est affiché à l'écran), on peut :

— *utiliser les commandes* que nous avons déjà utilisées dans le chapitre VIII (RUN, LIST, CLEAR, NUMB, LOAD et SAVE) ;

— *entrer de nouvelles instructions* (une étiquette suivie d'une instruction) ;

— *corriger une instruction* (en frappant l'étiquette ancienne suivie de l'instruction) ;

— mais aussi *utiliser l'ordinateur* en « MODE de BUREAU », c'est-à-dire en exécution immédiate :

1. CALCULETTE ALGEBRIQUE *

PRINT suivi d'une expression arithmétique**écrite sur la même ligne, entraîne l'évaluation de l'expression et l'affichage du résultat :

```
COMMAND :
PRINT 234 * 456 (RC)
106704
COMMAND :
```

Attention, nous n'écrirons plus (RC) en fin de ligne

— le mot PRINT peut être abrégé en PR ;
— l'expression arithmétique doit tenir sur la ligne ;
— les opérateurs arithmétiques s'écrivent " + " pour l'addition, " – " pour la soustraction, " * " pour la multiplication et " / " pour la division. Mais attention, aucun d'eux ne doit être sous-entendu :

3 (4 + 5) s'écrit 3 * (4 + 5)

On peut réaliser toutes les fonctions arithmétiques classiques suivantes :

SQR (....) : Racine carrée de (....)
ABS (....) : Valeur absolue de (....)
INT (....) : Partie entière de (....)

et si vous disposez du BASIC S : (S pour scientifique)

SIN (....) : Sinus d'un angle exprimé en radian
COS (....) : Pour le Cosinus
LOG (....) : Logarithme Népérien
EXP (....) : Exponentielle, *e* puissance (....)

Essayez quelques calculs utilisant ces fonctions et toute expression arithmétique... correctement écrite.

(*) *Calculette algébrique :* Calculette utilisant la notation algébrique des expressions arithmétiques : $(3+x)*(2+y)$, alors que d'autres utilisent d'autres notations, par exemple : $(3x+2y+*)$ pour définir l'exécution de la même expression.

(**) On appelle *expression arithmétique* toute expression comportant des constantes et/ou des variables séparées par des signes opératoires et/ou des parenthèses obéissants aux règles classiques de l'algèbre (les variables arithmétiques non définies au préalable sont considérées comme valant 0).

```
COMMAND :
PR ABS (- 3.14)
3.14

COMMAND :
PR INT (3.14)
3

COMMAND :
PR INT (- 3.14)
- 4
```

***Attention :* la partie entière de x est le plus grand entier inférieur ou égal à x !**

```
COMMAND :
PR 2/3
.66666667
```

On remarquera que les résultats s'écrivent avec 8 chiffres au plus, arrondis au dernier supérieur.

Enfin, les expressions s'écrivent sur une seule ligne et donc on doit faire très attention à utiliser les parenthèses à bon escient, et par exemple :

```
COMMAND :
PR (-5 +SQR (5 * 5-4 * 7 * (-3) ) ) / (2 * 7)
. 38859336
```

Pour calculer la racine $x = \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$

de l'équation $7x^2 + 5x - 3 = 0$

2. QUELLES DONNEES NUMERIQUES SONT ACCEPTEES PAR BASIC ?

On peut essayer de faire afficher le résultat de toute expression arithmétique donnant un résultat très grand ou très petit, positif ou négatif, mais un petit programme ira plus vite. Essayez le programme suivant :

```
10  X = 1
20  X = X * 10
30  PRINT X
40  GOTO 20
RUN
```

Vous constaterez que les nombres entiers sont acceptés dans leur écriture normale tant qu'ils n'ont pas plus de 8 chiffres en BASIC N8, et 12 chiffres en BASIC N12, puis qu'ils s'affichent en notation dite « exponentielle », la lettre E suivie d'un Z signifiant 10 puissance z.

Exercices :

1. En utilisant le même programme (quelque peu modifié), calculez les puissances successives de 2 (la solution est donnée à la fin de ce chapitre)

2. Effectuez les mêmes calculs en utilisant des divisions successives pour obtenir les puissances négatives de 2.

N'oubliez pas que l'on peut arrêter l'exécution d'un programme par CTRL + C.

Que constatez-vous ?

```
2 PUISSANCE -212 = 1.5192938 E -64
2 PUISSANCE -213 = 0
2 PUISSANCE -214 = 0
.                .   .
.                .   .
.                .   .
.                .   .
```

Toute valeur numérique inférieure à 1 E −64 est considérée comme nulle ; donc, en Basic, les valeurs numériques que l'on peut manipuler seront bornées pour les nombres positifs, à :

$$1E\ -64 \leq x \leq 9.999\ 9999\ E\ +62$$

De plus, ils seront écrits sous forme entière ou décimale si 8 chiffres suffisent et sous forme exponentielle dans les autres cas, par exemple :

Essayer :

```
PRINT  2/3
PRINT  428/100
PRINT  99999999999999
```
(constatez l'arrondi)

Vous pouvez répéter les mêmes exercices avec les nombres négatifs.

3. ET QUELLES DONNEES ALPHANUMERIQUES ?

Rappelons que l'on peut définir des constantes alphabétiques (ou chaîne de caractères) en plaçant une suite de caractères entre guillemets et qu'ainsi, en mode de bureau, on peut faire afficher des libellés :

```
PR " BONJOUR " ; "    " ; " MADAME "
BONJOUR  MADAME

COMMAND :
PR " ABCDEFGHIJKLMNOPQRSTUVWXYZ "
ABCDEFGHIJKLMNOPQRSTUVWXYZ

COMMAND :
```

Remarquons qu'il n'y a pas de limitation de longueur " ; " séparent les différents libellés à afficher. Mais ce n'est guère intéressant tant qu'il n'y a pas de traitements effectués.

4. UTILISATION DU « MODE BUREAU » SUR UN PROGRAMME

La caractéristique du *mode bureau* est d'exécuter immédiatement une instruction non précédée d'une étiquette dès que l'on frappe le RETURN.

Aussi peut-on utiliser l'ordre PRINT pour connaître les valeurs finales des variables à l'issue de l'exécution d'un programme et, ainsi, vérifier le bon (ou constater le mauvais) déroulement du programme.

Tapez dans l'ordre :

```
CLEAR
PRINT X
10  X = 3
20  PRINT X * X
30  Z = 2 * X
RUN
PR X ; Z
et constatez !
```

On peut de même interrompre un programme en cours (par CTRL + C), puis interroger les variables par PRINT et relancer l'exécution de la suite du programme par la commande RESUME.

> *ATTENTION !*
>
> *On peut ainsi faire exécuter toutes les instructions Basic, mais certaines sont dangereuses car elles peuvent modifier l'état des variables du programme. Seule* PRINT *est sans danger aucun.*

5. TEMPS PERDU NE SE RATTRAPE GUERE

L'exercice suivant est fort classique dans les classes de l'école primaire, mais trompe encore bon nombre de gens.

« Soient deux villes A et B, distantes de 30 km. Un cycliste habitué parcourt cette distance en 1 h 30 mn, à la vitesse moyenne de 20 km/heure. Un récent tremblement de terre à transformé la route en un gigantesque dos d'âne parfaitement symétrique.

Notre cycliste ira moins vite à la montée et devra donc aller plus vite à la descente pour respecter son horaire habituel. Etant donné une vitesse moyenne V1 (en km/h) à la montée, quelle doit être sa vitesse moyenne à la descente pour respecter son horaire ? »

Les constantes de ce texte sont donc :

- D = 30 km, distance des deux villes ;
- V = 20 km/h, vitesse moyenne de parcours ;
- T = D/V (soir 1 h 30), temps de parcours habituel.

Qu'advient-il ?

La première partie (soit D/2) est maintenant parcourue à V1 km/h, le temps de parcours T1 sera donc de :

$$T1 = (D/2) / V1$$

et l'on souhaite que l'horaire soit respecté, donc la deuxième partie du trajet devra être couverte en :

$$T - T1$$

La vitesse du cycliste doit donc être de

$$V2 = (D/2) / (T - T1)$$

Programmons cet exercice : on constate que l'on va manipuler deux sortes de données :

— D, V et T, qui restent constantes au cours de cet exercice (et telles que les valeurs de deux d'entre elles auraient suffi).

— V1, T1 et V2 qui seront variables et dépendront de V1 (et des constantes) :

```
T1 = (D/2) / V1
V2 = (D/2) / (T-T1)
```

Parmi ces trois variables, une seule doit être donnée par l'utilisateur : V1 ; T1 n'est qu'un intermédiaire utilisé dans le calcul et V2 est le résultat final souhaité.

5.1. Instruction INPUT

L'entrée de donnée est effectuée par l'exécution de l'instruction INPUT, ce mot clé étant suivi du nom d'une ou plusieurs variables (séparées par des points-virgules) auxquelles seront affectées les valeurs frappées successivement au clavier.

Elle peut comporter un texte (écrit entre guillemets) qui remplacera le point d'interrogation, ainsi qu'on l'a vu au chapitre VIII. Ici, donc, la saisie de la valeur de V1 s'écrira :

```
10 INPUT " VITESSE MOYENNE A LA MONTEE ? : " ; V1
```

Frappez cette ligne, puis la commande RUN : l'ordinateur attend que l'on donne une valeur numérique au clavier (suivie d'un Return).

Vérifiez (en mode bureau) la valeur V1.

5.2 Affectation

Les valeurs V1, D (30) et T (1.5) permettent maintenant de calculer T1 puis V2, à l'aide de deux instructions d'affectations qui donnent aux variables T1 puis V2 les valeurs calculées des expressions écrites à droite du signe égal :

```
20 T1 = (30/2) / V1
30 V2 = (30/2) / (1.5-T1)
```

Remarque

On aurait, bien entendu, pu effectuer (30/2) et écrire :

```
20  T1 = 15 / V1
```

mais on peut aussi bien conserver un « identificateur », c'est-à-dire un nom pour les constantes et écrire :

```
20  T1 = (D/2) / V1
```

à la seule condition que la valeur 30 ait été affectée à l'identificateur D dans une instruction précédente, par exemple :

```
5  D = 30
```

Toute généralisation d'un programme est une bonne chose.

5.3. Instruction PRINT

Nous connaissons déjà l'instruction PRINT qui permet d'afficher des libellés et des valeurs des variables : ici

```
40  PRINT " VITESSE A LA DESCENTE : " ; V2
```

Un programme correctement écrit doit se terminer sur une instruction STOP, mais *lorsque la mémoire est vide d'autres instructions, l'ordinateur arrête l'exécution à la dernière.*

Essayez ce programme avec diverses valeurs de V1 (V1 = 20, V1 = 15, V1 = 11, V1 = 5, V1 = 10, ...) et critiquez-le.

Afin d'éviter d'avoir à frapper RUN *à chaque exécution, remarquez qu'il suffit d'utiliser l'instruction :*
50 GOTO 10
que nous avons déjà vue. L'arrêt du programme se fera par CTRL et C.

— Que se passe-t-il pour V1 = 10 ?
— Que se passe-t-il pour V1 < 10 ?

L'instruction conditionnelle IF nous permettra d'éviter cela.

Exercice 3

On peut souhaiter « entrer » des valeurs différentes pour les constantes D et T à chaque exécution : construire le programme qui le permet. (Solution en fin de chapitre.)

6. BONJOUR MONSIEUR, BONJOUR MADAME

Pour dire correctement « bonjour » à un homme ou une femme il suffit de connaître son numéro de sécurité sociale : celui d'un homme commence par le chiffre 1 et celui d'une femme par 2.

6.1. Chaînes de caractères

Une chaîne de caractères peut être « constante » et intervenir dans un ordre PRINT, par exemple :

```
10  PRINT " BONJOUR "
RUN
```

ou dans un ordre INPUT (pour préciser la nature de la donnée attendue).

Mais elle peut être variable et son nom doit être alors suivi du symbole « $ ». Un « identificateur » de chaîne alphanumérique s'écrira donc : une lettre, suivie éventuellement d'un chiffre, suivi du symbole $.

Si rien n'est précisé, la longueur de la chaîne sera de 10 caractères en mémoire (quelle que soit la longueur frappée au clavier).

Si l'on désire mémoriser *n* caractères, on doit « déclarer » la chaîne avant son emploi par l'instruction :

DIM Id$ (n) (Id est l'identificateur)

Dans cet exercice, acceptons les noms propres de 20 caractères et désignons par A$ l'identificateur du nom propre :

```
10 DIM A$ (20)
```

> ***Attention !***
>
> *Avez-vous utilisé la commande* CLEAR *pour effacer le programme précédent ?* Vérifiez en frappant LIST.

```
20 INPUT " QUEL EST VOTRE NOM PROPRE ? : " ; A$
```

Quel est votre numéro de sécurité sociale ?

— c'est un nombre à 13 chiffres... que Basic va transformer en notation exponentielle (en nous perdrons les 5 derniers en Basic 8... !).

— c'est une chaîne de 13 caractères (13 chiffres) dont nous voulons extraire le 1er. Soit S$ son identificateur :

```
30 INPUT " QUEL EST VOTRE NO S.S. ? : " ; S$
```

Remarquons que l'absence de dimensionnement limite **S$** *aux 10 premiers caractères frappés au clavier : essayez* **PRINT S$.**

On doit extraire le 1er caractère et, pour cela, utiliser une :

6.2 Sous-chaîne

On appelle sous-chaîne Y$ d'une chaîne X$, un sous-ensemble des caractères de X$ défini par les indices de début et de fin de ce sous-ensemble :

$$Y\$ = X\$ (I;J)$$

Y$ est un sous ensemble des caractères de la chaîne X$ du Ie au Je compris (avec $I \leqslant J$)

Exemple :

```
X$ = " BONJOUR "
Y$ = X$ (3;5)
PRINT Y$
NJO
```

Essayez en mode de bureau **(c'est-à-dire sans numéro d'instruction devant) ;** *chaque instruction est exécutée au fur et à mesure.*

Exemple : X$ (I;I) désigne une sous-chaîne contenant le Ie caractère de X$.

6.3. L'instruction IF

Nous devons exécuter une certaine instruction (PRINT " BONJOUR MR. " ; A$) si le 1er caractère de S$ est un 1 et une autre (PRINT " BONJOUR ME. " ; A$) si c'est un 2, donc disposer d'une instruction « conditionnelle » : c'est le IF qui s'écrit (en toute généralité) :

IF *une condition* THEN *une instruction* ELSE *une autre instruction*

— L'instruction suivant le THEN est exécutée si la condition est vérifiée ; sinon, c'est celle qui suit le ELSE qui l'est.

— La condition qui suit le IF peut être une égalité ou une inégalité entre constantes, variables ou expressions arithmétiques, ou encore constantes ou variables alphanumériques (l'inégalité sera alors prise au sens de l'ordre alphabétique des mots).

— Enfin, le ELSE et l'instruction qui suit peuvent être omis et dans ce cas, l'instruction suivante est exécutée à sa place.

On peut donc écrire :

```
40 IF S$ (1;1) = "1 " THEN PR " BONJOUR MR. " ; A$ ELSE PR " BONJOUR ME. " ; A$
```

Mais il faut que toute l'instruction tienne dans la ligne.

Très souvent, on utilisera l'instruction GOTO *e* (aller à l'instruction *e*) pour faire exécuter un bloc d'instruction dans l'un ou l'autre cas.

Mais l'écriture de l'instruction 40 n'est pas satisfaisante car elle suppose que si le 1er caractère n'est pas un « 1 », alors c'est un « 2 » ; or, on peut très bien avoir frappé autre chose, ou même un blanc qui n'apparaît pas à l'écran.

Exercice 4

• *Compléter le programme à des fins de vérification du 1er chiffre du numéro de sécurité sociale.* (Solution en fin de chapitre.)

• *Peut-on vérifier la validité des autres éléments ? (Ne pas oublier de dimensionner à 13 pour les mémoriser tous et demandez à ce qu'ils soient frappés sans blancs ni séparateurs.)*

Exercice : Vous pouvez maintenant revenir sur l'exercice 3 et effectuer tous les tests voulus sur les données et les résultats :

— éviter une division par zéro dans l'instruction 30 ;
— éviter un résultat négatif si T1 > T ;
— vérifier que D et T sont positifs et non nuls ; etc.

***Ceci est très important* car, lors d'une exécution « à la main », vous faites toutes ces vérifications sur les valeurs que vous obtenez tandis que dans un programme, elles doivent être toutes prévues.**

7. PARQUEZ VOS MOUTONS

Un éleveur dispose d'un terrain de pâture situé en bordure d'une voie autoroutière protégée par une forte clôture. Il dispose, par ailleurs, d'une longueur P d'éléments de clôture mobile en segments de 1 m. Il souhaite réaliser un champ d'élevage rectangulaire de surface maximale. Quelles sont les dimensions idéales ?

Soient L1 et L2 les côtés et S la surface du rectangle (fig. 11-1). Remarquons, tout d'abord, que P doit être supérieur ou égal à 3, que L2 est borné par P lui-même (un champ de surface nulle collé à l'autoroute) et que L1 est borné par la partie entière de P/2.

Il est tentant de calculer toutes les surfaces possibles obtenues en donnant à L1 (ou L2) toutes les valeurs possibles et de retenir la solution maximale :

• L1 peut varier de 0 à INT* (P/2) ;
• L2 vaut alors P − 2 * L1 ;
• et S = L1 * L2, soit S = L1 * (P − 2 * L1)

(*) La partie entière de X se note INT (X).

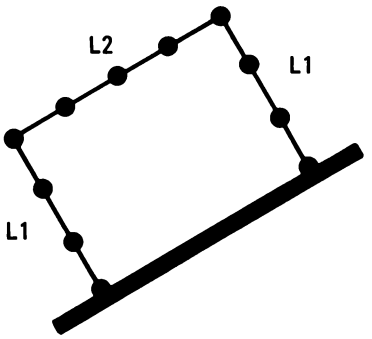


Fig. 11.1 — « Parquez vos moutons » : voici l'aspect du champ d'élevage.

7.1. L'itération : la boucle de 1 à N

Tout calcul répétitif dont on connaît les bornes peut s'exécuter grâce à l'instruction FOR :

```
e    FOR I = 1 TO N
e'   inst 1
     ...
     inst j
e''  NEXT I
```

L'ensemble des instructions comprises entre l'instruction FOR et l'instruction NEXT I seront exécutées N fois, I prenant les valeurs successives 1, 2, 3, ...N – 1, N, après quoi le programme continuera à l'instruction suivant NEXT I. Ici donc, I varie de 1 à INT (P/2) + 1, L1 valant I – 1 dans chaque calcul. Une solution serait donc :

```
10 INPUT " VALEUR DE P ? : " ; P
20 FOR I = 1 TO INT (P/2)+1
30 L1 = I-1 \L2 = P-2 * L1
40 S = L1 * L2
50 PR L1 ; "^^ " ; L2 ; "^^" ; S
60 NEXT I
70 STOP
```

Le séparateur permet de mettre plusieurs instructions sur la même ligne et 1 seule étiquette.

On améliore nettement la présentation en disposant les résultats en tableau grâce à TAB *(n)* qui, dans l'ordre PRINT, positionne l'affichage à la colonne *n :*

```
50  PR L1 ; TAB (7) ; L2 ; TAB (14) ; S
```

et en précédant le tout d'un en-tête :

```
15  PR " ∧ L1 ∧∧∧ L2 ∧∧∧∧ S "
16  PR " **** ∧∧ **** ∧∧∧ *** "
```

dans le texte imprimé « ∧ » signifie « laisser un espace ».

7.2. Recherche d'un maximum

On peut laisser à l'opérateur le soin de lire la valeur maximale dans son tableau, mais aussi chercher à mémoriser la valeur maximale de S (et les valeurs correspondantes de L1 et L2) au fur et à mesure des calculs dans la boucle.

En effet, en posant M = 0 avant tout calcul et en mémorisant dans M la plus grande des deux valeurs, S calculée et M (maximum précédent) on obtiendra la valeur maximale de tous les S calculés.

Exercice 5

Modifiez le programme précédent pour trouver la valeur maximale et une solution (L1, L2) ? (Solution en fin de leçon.)

- *Pourquoi une solution ?*
- *Peut-il y en avoir deux ?*

Remarque sur la nature mathématique du problème. - En supposant la variable L1 continue (nous l'écrirons x par habitude), la surface S est une fonction de x avec :

$$S(x) = x(P - 2x)$$

et, s'il existe un extremum, ce sera pour la valeur de x annulant la dérivée :

$$S'_x(x) = P - 4x$$

On vérifie facilement que $x = P/4$ donne un maximum (encore faut-il calculer les valeurs de S obtenues pour les entiers L1 encadrant P/4 si P n'est pas un multiple de 4).

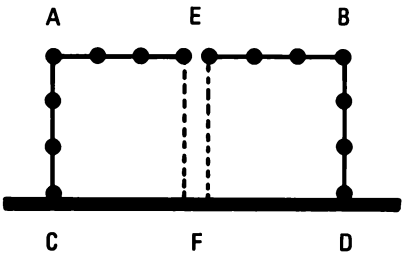


Fig. 11.2 — Une autre façon de trouver le maximum.

Remarque de bon sens. - Si on coupe AB en deux parties égales (fig. 11-2), on constate que notre problème revient à maximiser la surface d'un rectangle AEFC dont un demi-périmètre (EA + AC) est donné. La solution fort connue est un carré. On retrouve donc facilement le maximum de S pour L1 = P/4.

7.3. Généralisation de la boucle FOR BASIC

On peut écrire une instruction de « boucle » dont la variable ne soit ni entière, ni même croissante à l'exécution :

n FOR *var* = *exp*1 TO *exp*2 STEP *exp*3
m NEXT *var*

L'expression qui suit le mot STEP est évaluée et définit le « pas d'incrémentation » de la variable à chaque exécution du « corps de la boucle ». L'exécution de la boucle cesse lorsque la variable incrémentée sort de la fourchette définie par [*exp*1, *exp*2], bornes comprises.

Le pas peut être entier ou non, positif ou non. Si STEP *exp*3 est omis, le pas est pris par défaut égal à 1.

Exercice :

Faites imprimer le carré des nombres pairs en ordre décroissant de 36 à 20.

8. TEMPERATURE DE L'EAU DU ROBINET

Il est fréquent que, le matin et dans les étages élevés d'un immeuble, l'ouverture du robinet d'eau chaude ne donne que de l'eau froide, puis tiède, puis... chaude. Aussi le bain risque-t-il d'être froid, puis tiède, puis... Comment calculer la température du bain connaissant la température de l'eau du robinet.

Les physiciens disent que pour calculer la température moyenne Tm d'un mélange de Q1 litres d'eau à T1 degrés et de Q2 litres à T2 degrés, il suffit de calculer :

$$Tm = \frac{Q1 \times T1 \times + Q2 \times T2}{(Q1 + Q2)}$$

Mais que se passe-t-il si la température évolue de façon continue à la sortie du robinet ?

En première approximation, nous allons relever la température de l'eau toutes les secondes (voire, toutes les 5 ou 10 si l'eau chaude est vraiment longue à venir). On obtiendra une *table de valeurs :* $t_0 = 12$, $t_1 = 12$, $t_2 = 12{,}5$, $t_3 = 13$, $t_{n-1} = 71$, $t_n = 72$, ou un graphe de la forme de la figure 11-3.

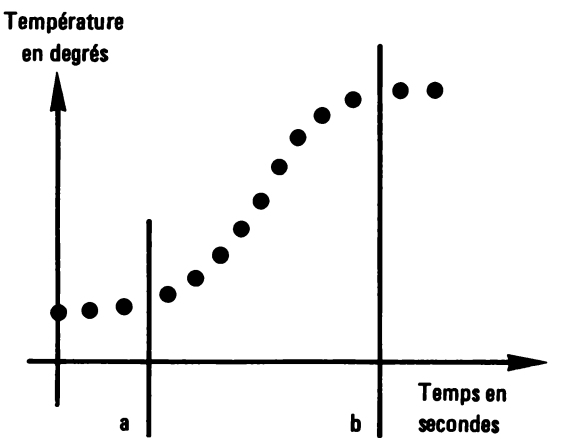


Fig. 11.3 — Graphe des températures.

En généralisant la formule précédente et en considérant que la température de l'eau qui coule varie peu durant une seconde (précision qui, pour un bain, est fort suffisante), on obtient comme température moyenne de l'eau recueillie entre les secondes *a* et *b* :

$$Tm = \frac{q \times (a) + q \times t(a+1) + q \times t(a+2) \ldots + q \times t(b-1)}{q + q + q \ldots + q}$$

Nous avons supposé le débit constant et égal à q litres par secondes (mais il va de soi que l'on peut généraliser si on a relevé les différents débits en fonction du temps). Cette formule se simplifie vite :

$$T_m = \frac{t(a) + t(a+1) + t(a+2) + ... + t(b-1)}{b - a}$$

8.1. Notion de tableau

Lorsqu'on travaille sur quelques relevés (3 ou 4), on peut encore leur donner des *identificateurs* différents X, Y, Z, T ou même T1, T2, T3, T4, et la formule précédente deviendrait l'expression suivante :

$$T = (X+Y+Z+T) / (B-A)$$

Mais dans le cas d'un grand nombre de données, nous serions conduits à utiliser un grand nombre d'identificateurs * et les expressions de calcul arithmétique seraient lourdes à écrire.

Or, Basic (et tous les autres langages de programmation) utilise la notion de *tableau.* L'ensemble des valeurs reçoit *un nom* et chacune d'elles est précisée par un *indice (son rang)* placé entre parenthèses. Pour des températures, prenons l'identificateur T. Nos relevés seront donc notés en Basic : T(0), T(1), T( )..., T(N).

> *Attention !*
> Un indice est un nombre entier qui, en *Basic LX 500,* varie de *0* à *N (N, indice maximum du tableau de N+1 données).*

Le seul fait qu'un identificateur soit suivi de parenthèses signale la nature « TABLEAU » de l'identificateur, mais sans spécification supplémentaire, l'indice varie de 0 à 10.

Un tableau de taille supérieure (et même, à notre avis, *TOUT TABLEAU*) doit être déclaré par l'instruction DIM suivie de l'identificateur du tableau et du rang maximum de l'indice entre parenthèses.

***Tout dépassement sera signalé à l'exécution par :* RANGE ERROR.**

(*) *Un petit calcul :* En Basic, un identificateur s'écrit avec une lettre éventuellement suivie d'un chiffre. Il y en a donc .... $26+26\times10 = 286$ au plus de disponibles.

```
5 DIM Z (20)
```

Z contiendra 21 valeurs Z(0), Z(1), ... Z(20)

> *Attention !*
>
> *Le rang maximum peut être lui-même une variable, à la seule condition qu'elle ait été définie avant.*

L'expression suivante est ainsi correcte :

```
3 K = 20
5 DIM Z(K)
```

L'intérêt serait mince si les indices étaient des constantes (car on devrait écrire toutes les valeurs de l'indice dans les expressions de traitement). Mais la notation T(I) est valable si I a été défini au préalable.

> *Attention !*
>
> A chaque *RUN,* le Basic du *LX 500* initialise les variables numériques à 0 lors de la première apparition. *Donc, si I n'a pas été initialisé au préalable, la valeur T(I) sera prise comme T(0) sans diagnostic d'erreur.*

Aussi, la notion de *boucle* précédemment définie et l'utilisation des tableaux vont nous permettre de résoudre beaucoup des problèmes à structure répétitive ; en particulier, *la somme des éléments d'un tableau.*

8.2. Calcul de s = t(a) + t(a+1) + ... + t(b−1)

Appelons s_i la somme des i premiers termes.

$$s_0 = 0$$
$$s_1 = s_0 + t(a)$$
$$s_2 = s_1 + t(a+1)$$
$$s_i = s_{i-1} + t(a+i-1)$$
$$\ldots \qquad \ldots$$
$$s = s_{b-2} + t(b-1)$$

Nous avons déjà rencontré au chapitre 8 la notion d'accumulateur, c'est-à-dire d'une variable dont la valeur sera, à chaque fois, augmentée d'une valeur quelconque. Elle mémorise ainsi le total partiel de ces valeurs. L'écriture Basic est alors simple :

```
10  S = 0
20  FOR J = A TO B-1
30  S = S + T(J)
40  NEXT J
```

Il vous reste, bien entendu, à terminer ce programme en entrant des valeurs de T, les bornes A et B et à imprimer le résultat : à vos claviers..., mais avant, à vos plumes.

Même pour un petit programme, il est très recommandé de l'étudier et de le concevoir sur papier avant de commencer à l'enregistrer !

9. PLURIEL DES NOMS EN « OU »

Règle

Les noms communs se terminant en « ou » prennent un « S » au pluriel, sauf BIJOU, CAILLOU, CHOU, GENOU, HIBOU, JOUJOU et POU qui prennent un X.

Problème - Un mot se terminant par « ou », étant donné, écrire un programme qui affiche son pluriel correct. Qu'avons-nous à faire ?

1. Tout d'abord, vérifier qu'il se termine par « OU », c'est-à-dire, isoler les deux derniers caractères en une sous-chaîne, et la comparer à « OU ».

2. Puis, chercher si le mot donné appartient à la liste des exceptions.

3. Enfin, et suivant le cas, afficher le mot donné suivi d'un S ou d'un X.

9.1. Travail sur sous-chaînes

Nous avons déjà rencontré des sous-chaînes et extrait une sous-chaîne d'une chaîne :

```
S$ = C$(I;J)
```

S$ est une chaîne formée des caractères de C$ compris entre le I^e^ et le J^e^ rang.

— I doit être inférieur ou égal à J.

— I et J doivent être compris dans les bornes [1,N] (où N est la longueur de la chaîne).

***Remarque :* Si I et J ne sont pas entiers, leur valeur entière donne le rang (attention : 3*(4/3) donnera 3 pour 3.9999999 et 3*(5/3) donnera 5 pour 5.0000001 !**

Mais il nous faut connaître parfaitement la dimension de la chaîne, au moment de l'extraction, pour que les indices I et J soient corrects.

***En effet, rappelons qu'une chaîne de caractères est « dynamique », c'est-à-dire que sa longueur peut varier au cours de l'exécution du programme* (entre 1 et N si l'on a dimensionné la chaîne à N caractères, entre 1 et 10 sinon).**

On lèvera ce problème de deux façons :

- la première en omettant l'indice J :

$$A\$ = C\$(I)$$

où A$ est une sous-chaîne comportant les caractères de C$ à partir du I^e^ compris et jusqu'à la fin ; mais cela exige encore que I soit correct, donc $1 \leq I \leq N$;

- la seconde en utilisant la longueur de la chaîne :

$$L = LEN(C\$)$$

La fonction LEN donne la longueur effective de la chaîne C$.

On pourra alors utiliser cette valeur L soit pour vérifier que I et J sont corrects et compris entre 1 et L, soit même pour calculer les indices I et J eux-mêmes.

Exercice 6

Quels sont les indices des deux derniers caractères de la chaîne M$. (Solution en fin de chapitre)

Exercice 7

Ecrire un programme qui vérifie si un mot donné se termine par OU. (Solution en fin de chapitre)

On utilise la même technique pour voir si un verbe se termine en ER, et plus généralement, pour toutes les questions de suffixe.

9.2. RETROUVER UN MOT DANS UNE CHAINE

On aura très souvent besoin de retrouver un mot dans une liste donnée (ou bien de vérifier sa présence comme ici, ou encore de déterminer son rang dans cette liste). Ce traitement est relativement facile à effectuer sur des nombres, rangés dans un tableau, puisqu'on peut « indicer » chaque nombre et « balayer » le tableau (à l'aide d'une boucle par exemple).

Exercice 8

Soit un tableau A, contenant 10 valeurs numériques, écrire un programme qui vérifie la présence d'un nombre X donné. (Solution en fin de chapitre)

Malheureusement, en Basic, on ne peut pas placer de chaînes numériques dans un tableau et donc, on ne peut pas indicer les mots d'une liste *. On pourrait, bien entendu, écrire autant d'instructions de comparaisons qu'il y a de mots dans la liste, mais, si cela est possible pour les 7 exceptions du pluriel en OU, c'est impossible lorsque la longueur de la liste est, à priori inconnue, ou qu'elle est trop longue.

(*) Cette possibilité existe en LSE, niveau 2.

Le *CALCUL d'INDICE* va nous permettre de recréer la structure de tableau dans une chaîne unique.

Soit une chaîne U$; décidons que les mots sur lesquels nous allons travailler ont une longueur inférieure ou égale à 20 caractères. Nous allons découper (virtuellement) la chaîne U$ en morceaux de 20 caractères et les numéroter 1, 2, ..., I, III (fig. 11.4).

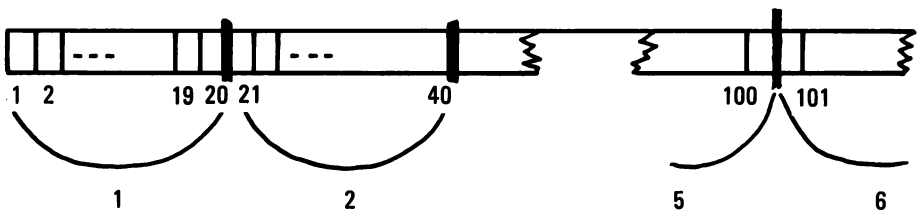


Fig. 11.4 — Découpe virtuelle de la chaîne en sections numérotées de 20 caractères chacunes.

Exercice 9

Si chaque caractère de U$ est numéroté de 1 à ...N (longueur de U$), quels sont les valeurs des bornes :

- *du 1er mot 1 et 20,*
- *du 2e mot ... et ...,*
- *du 3e mot ... et ...,*

...

- *du mot de rang X ... et ...*

(Solution en fin de chapitre)

Pour manipuler les valeurs des éléments d'un tableau, un indice suffisait pour désigner l'élément ; ici *le rang* (l'indice, en quelque sorte) *du mot* nous permet d'extraire la sous-chaîne qui le contient :

1er mot : U$ (1;20)
2e mot : U$ (21;40)
mot X : U$ ((X – 1) * 20 – 1 ; X * 20)

Attention !

$$1 \leqslant X \leqslant \text{INT } (N/20)$$

INT : fonction partie entière ;
N : longueur de la chaîne supérieure ou égale à 20

Le mot « TOTO » est-il dans la chaîne ? Il suffit de sortir les mots un par un et de les comparer à « TOTO ». Mais TOTO n'a que 4 caractères et les sous-chaînes, les mots extraits de la liste en ont 20 ! Or, dans la mémoire d'un ordinateur, tous les caractères (même les blancs) sont significatifs dans une comparaison : « TOTO » de longueur 4 est différent de « TOTO » de longueur 5, à cause du caractère blanc qui, (bien qu'invisible à l'écran), existe dans la chaîne mémorisée.

Une solution : compléter chaque donnée par un caractère spécial (un blanc, une étoile, un point...) non significatif dans l'alphabet, pour compléter à 20 caractères.

Le caractère « blanc » *est gênant car il est invisible à l'écran et notre œil a trop l'habitude de ne lire que les caractères affichés.*

Le « point » *est agréable car il permet de suivre la ligne sur l'écran.*

Exercice 10

Soit un mot X$ de longueur inconnue, le compléter par des étoiles pour obtenir un bloc de 20 caractères.

1° Quel nombre d'étoiles doit-on rajouter ?
(Solution en fin de chapitre)

Ajouter D étoiles revient à compléter X$ par une chaîne contenant D étoiles, ou encore, une sous-chaîne de longueur D d'une chaîne contenant 20 étoiles :

```
5   DIM E$ (20)
10  E$ = "**********"
         20 étoiles
    Y$ = X$ + E$ (1;D)
```

Cette dernière instruction introduit la *CONCATENATION* de deux chaînes, c'est-à-dire la *constitution d'une seule chaîne à partir de deux autres chaînes.*

La notation " + " utilisée par Basic ne doit pas prêter à confusion : il ne s'agit pas d'une opération arithmétique, mais d'un traitement lexical sur des chaînes de caractères. Ainsi, " 19 " + " 81 " donne " 1981 " !

Exemple - Si :

X$ = " XYZTUV "

L = LEN (X$) vaut 6, D vaut 14 ce qui conduit à :

Y$ = XYZTUV**********
20 caractères

Exercice 11

Soit une chaîne U$ de longueur N (N ≥ 20) constituée de blocs de 20 caractères comprenant chacun un mot suivi d'étoiles éventuelles, écrire un programme qui cherche si " TOTO " figure dans la chaîne. (Solution en fin de chapitre)

Vous êtes maintenant prêt à rédiger un programme qui écrive les noms en " OU " au pluriel.

9.3. Pluriel

9.3.1. Constitution de la chaîne des exceptions

Il faut constituer une chaîne U$ comportant 7 blocs avec les mots BIJOU, CAILLOU, CHOU, GENOU, HIBOU, JOUJOU et POU suivis d'autant d'étoiles que nécessaire pour faire des blocs de 20 caractères.

Nous avons vu que l'on peut constituer une chaîne à l'aide de sous-chaines en effectuant une « CONCATENATION » (symbole " + ") ; aussi n'aurez-vous aucune peine à suivre le petit programme suivant :

```
10  DIM U$ (140)
20  U$ = " BIJOU********** "
30  U$ = U$ + " CAILLOU********** "
40  U$ = U$ + " CHOU********** "
50  U$ = U$ + " GENOU********** "
60  U$ = U$ + " HIBOU********** "
70  U$ = U$ + " JOUJOU********** "
80  U$ = U$ + " POU********** "
```

Mais on peut aussi la construire en « conversationnel » :

```
10  DIM U$ (140) ; E$ (20) ; A$ (20)
20  E$ = " ********** "
30  U$ = " "
40  FOR I = 1 TO 7
50  PR " EXCEPTION NO     " ; I ; " ^ :     " ;
60  INPUT "  ^  " ; A$
70  U$ = U$ + A$ + E$(1 ; 20 - LEN(A$) )
80  NEXT (I)
```

En guise de vérification, vous pouvez faire exécuter les deux programmes suivis de :

```
90  PR U$
```

Remarques :

— Le point virgule, en fin d'instruction 50, laisse le curseur sur la même ligne : ainsi, l'entrée de la donnée se fera *sur la même ligne* que la question.

— Dans l'instruction 60, le libellé " ^ " remplacera le point d'interrogation qui ne fait pas beau dans la question (essayez avec 60 INPUT A$ et concluez).

9.3.2. Acquisition du mot M$

```
5   DIM M$ (20) ; N$ (20)
100 INPUT " QUEL EST LE MOT ? : " ; M$
110 IF M$ < >" " THEN 140
120 PR " FIN DU PROGRAMME "
130 STOP
140 L = LEN (M$)
150 IF M$ (L-1;L) = " OU " THEN 180
160 PR M$ ; " NE SE TERMINE PAS PAR OU. "
170 GOTO 100
180 IF L < = 20 THEN 210
190 PR M$ ; " EST UN MOT TROP LONG "
200 GOTO 100
210 IF L = 20 THEN 240
220 N$ = M$ + E$(1;20-L)
230 GOTO 300
240 N$ = M$
```

Remarques :

— Pourquoi N$? Nous avons besoin de conserver le mot au singulier pour le mettre au pluriel plus tard.

— En 180, pourquoi se demander si L est inférieur ou égal à 20 ? En effet, lors de la saisie de M$ au clavier, le mot frappé au clavier sera tronqué à 20 caractères au plus en fonction de l'instruction de dimensionnement. On devrait donc déclarer M à ... 80 (une ligne au plus) et l'instruction 180 prendra son effet :

```
5 DIM M$(80) ; N$(20)
```

— En 110, vous constaterez que la donnée d'un mot vide arrête le programme (pour cela, frappez directement *return).*

9.3.3. M$ est-il une exception ? ou N$ existe-t-il dans U$?

```
300  FOR I = 1 TO 7
310  IF N$ = U$ ( (I-1) * 20+1·; I * 20) THEN EXIT 350
320  NEXT I
330  M$ = M$ + " S "
340  GOTO 400
350  M$ = M$ + " X "
```

9.3.4. Affichage du pluriel

```
400  PR " SON PLURIEL EST " ; M$
410  GOTO 100
```

Malheureusement, le mot OU tout seul prend encore un S... et qu'une frappe de 2 lettres entraîne une erreur. Pourquoi ? Corrigez.

Vous constaterez que la relecture de ce programme n'est pas facile ; aussi nous vous conseillons de le *documenter* à l'aide de commentaires, de sous-titres, de lignes blanches, grâce à l'instruction (ou la pseudo-instruction) REM.

Voulez-vous le sauvegarder ? Souvenez-vous :

```
QUIT
**BDOS ...
? C ; PLUR ; 5 ; P
B
.
COMMAND :
SAVE PLUR
```

La taille déclarée ici (5 secteurs de disquette, soit 1 280 caractères, est bien grande, mais n'oubliez pas que vos commentaires prennent de la place.

10. CALCUL D'ENGRENAGES

Les mécaniciens et les cyclistes connaissent bien ce problème de « démultiplication » des engrenages. Comment combiner quelques roues (fig. 10.5) de

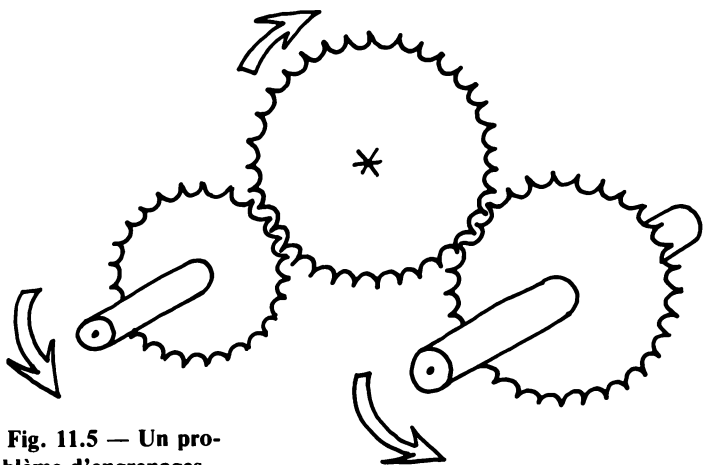


Fig. 11.5 — Un problème d'engrenages...

rayons R1, R2 et R3 pour obtenir un rapport donné x entre les vitesses des axes des roues 1 et 3 ? (Dans notre cas, x est positif... ; les roues tournent dans le même sens.)

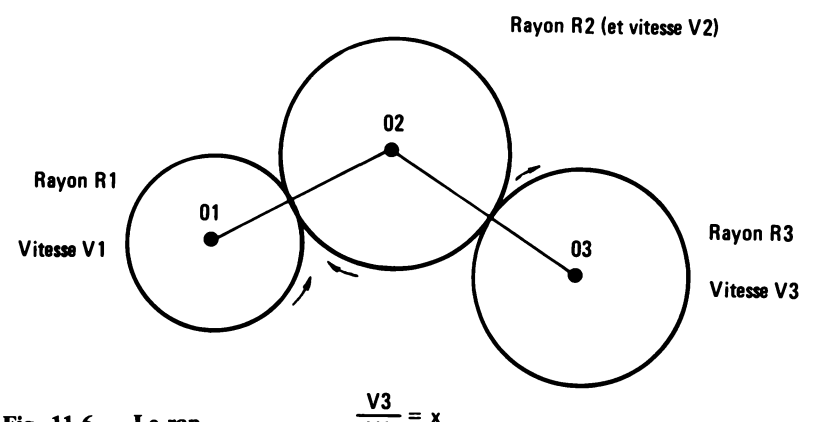


Fig. 11.6 — Le rapport x est donné par V3/V1.

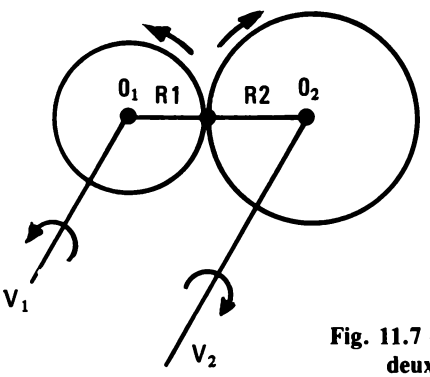


Fig. 11.7 — Système à deux roues.

Schématisons quelque peu (fig. 11.6). Les mécaniciens (et le bon sens) disent que dans un système de deux roues dentées, on a (fig. 11.7) : $V_1 \times R_1 = V_2 \times R_2$, ou

$$\frac{V_2}{V_1} = \frac{R_1}{R_2}$$

Donc :

$$\frac{V_2}{V_1} = \frac{R_1}{R_2} \text{ et } \frac{V_3}{V_2} = \frac{R_2}{R_3}$$

L'élimination de l'intermédiaire V_2 mène à

$$\frac{V_3}{V_1} = \frac{R_1}{R_3} = x$$

Il s'agit donc, pour nous, de trouver dans un stock de roues dentées, deux roues dont le rapport des rayons soit x, ainsi qu'une roue n° 2 permettant le transfert du mouvement.

Une rapide étude géométrique vous mènera aux contraintes suivantes : si d est la mesure de la distance O_1, O_2 (entraxe), R_1, R_2, R_3 doivent donc vérifier :

$$R_1 + 2 \times R_2 + R_3 \geqq d > R_1 + R_3$$

Donc, O_1 O_2 O_3 est un triangle dont les côtés sont $R_1 + R_2$, $R_2 + R_3$ et d, et les roues 1 et 3 doivent être extérieures l'une à l'autre.

Problème - Etant donné un ensemble de roues dentées (dentures compatibles) dont les rayons sont enregistrés dans un tableau R, quelles combinaisons donne un rapport de transmission x positif donné dans le montage précédent ? (Il va de soi que l'on dispose de plusieurs exemplaires de la même roue.)

Toutes les combinaisons sont *à priori* possibles ; donc, il nous faut essayer chacune des roues avec toutes les autres et elle-même. Ceci nous conduit à la structure de boucles emboîtées qui, ici, s'écrirait :

```
n1  FOR I = 1 TO N
n2  FOR J = 1 TO N
          ...
n3  NEXT J
n4  NEXT I
```

Exercice

Essayez le programme suivant et suivez-en le déroulement :

```
10  FOR I = 1 TO 4
20  FOR J = 1 TO 4
30  PR " I = " ; I ; " J = " ; J
40  NEXT J
50  NEXT I
60  STOP
```

(L'instruction 30 est effectuée pour toutes les combinaisons possibles des valeurs de I et J)

Notre problème nous conduit ainsi à essayer toutes les combinaisons possibles de rayons R1 et R3, à regarder si le rapport R3 sur R1 vaut x, puis à voir si les contraintes sont satisfaites.

Soit D l'entraxe, X le rapport souhaité et R le tableau contenant les valeurs des N rayons disponibles. Prenons I comme indice de boucle pour la première roue, J pour la 3e roue et K pour la seconde (en remarquant qu'elle n'intervient qu'au niveau des contraintes).

10.1. Recherche d'un couple R1, R3

```
110  FOR I = 1 TO N
120  FOR J = 1 TO N
130  IF R(J) / R(I) < > X THEN 150
140  PR R(I) ; " ET " ; R(J) ; " CONVIENNENT
150  NEXT J
160  NEXT I
170  STOP
```

Les instructions sont numérotées à partir de 100 car il faudra, au préalable, créer le tableau R et entrer X et D.

```
10  INPUT " NOMBRE DE TAILLES ?  " ; N
20  DIM R(N)
30  FOR I = 1 TO N
40  PR " RAYON NO  " ; I ;
50  INPUT " :  " ; R(I)
60  NEXT I
70  INPUT " QUEL RAPPORT DESIREZ VOUS ? " ; X
80  INPUT " QUEL EST L'ENTRAXE ? " ; D
```

Faites un premier essai.

— Constatez qu'à chaque exécution, il faut entrer à nouveau le tableau des rayons. Alors remplacez :

```
170  STOP par 170  GOTO 70
```

— Il n'y a pas toujours de solution. Nous vous proposons de réfléchir à un programme qui calculerait alors le meilleur jeu d'engrenages. *(La solution de ce 13e exercice est donnée en fin de chapitre)*

10.2. Vérification des contraintes

Nous devons vérifier deux contraintes dès qu'un couple R1, R3 donne un rapport K. La première porte sur R1 et R3 : D > R1 + R3, et si elle est satisfaite, la seconde : R1 + 2R2 + R3 ≧ D exige la recherche d'une roue R2 satisfaisant cette condition, donc un nouveau balayage de la table R. Voici une solution :

```
110  FOR I = 1 TO N
120  FOR J = 1 TO N
130  IF R(J) / R(I) < > X THEN 150
133  IF D < = R(I) + R(J) THEN 150
135  FOR K = 1 TO N
137  IF R(I) + 2 * R(K) + R(J) < D THEN 143
140  PR R(I) ; R(K) ; R(J) ; " CONVIENNENT. "
143  NEXT K
150  NEXT J
160  NEXT I
170  GOTO 70
```

On obtient encore beaucoup de solutions avec plusieurs valeurs de R2 pour le même couple de roues R1 et R3. Nous vous proposons d'automatiser la recherche en cherchant la combinaison donnant trois roues de rayons équilibrés. *(La solution de ce problème se trouve également en fin de chapitre)*

Enfin, on prévoit une instruction d'arrêt qui stoppe le programme si le rapport demandé est nul, par exemple :

```
75  IF X = 0 THEN STOP
```

Remarque **- Ce programme comporte des divisions ; donc on doit se protéger des divisions par zéro. Une erreur dans les données est possible ! :**

```
125  IF R(I) = 0 THEN 150
```

Remplacez la roue 2 par une chaîne et embarquez un ordinateur sur votre bicyclette !

SOLUTION DES EXERCICES

Exercice 1

```
10  N = 0  \  X = 1
20  N = N + 1  \  X = X * 2
30  PRINT " 2 ∧ PUISSANCE ∧ " ; N ; " ∧ = ∧ " ; X
40  GOTO 20
RUN
```

```
2  PUISSANCE 1    = 2
2  PUISSANCE 2    = 4
...               ...     ...
...               ...     ...
2  PUISSANCE 26   = 67108864
2  PUISSANCE 27   = 1.3421773 E + 08
...               ...     ...
...               ...     ...
2  PUISSANCE 209  = 8.2275208 E + 62
NUM OVERFL ERROR IN LINE : 20
COMMAND
```

Le symbole \ permet de placer plusieurs instructions sur la même ligne.

Exercice 2

Il a été développé dans le chapitre.

Exercice 3

```
1   REM * TEMPS PERDU NE SE RATRAPPE GUERE *
3   INPUT " DISTANCE DES 2 VILLES EN KM ? " ; D
5   INPUT " TEMPS MOYEN ORDINAIRE ? " ; T
10  INPUT " VITESSE A LA MONTEE ? : " ; V1
20  T1 = (D/2) / V1
30  V2 = (D/2) / T-T1)
40  PRINT " VITESSE A LA DESCENTE : " ; V2
50  GOTO 10
```

L'instruction 1 REM * ... ne sert qu'à donner des informations sur le programme (un titre, une explication, ...) mais n'a aucune influence à l'exécution.

Exercice 4

```
10  DIM A$ (20)
20  INPUT " QUEL EST VOTRE NOM PROPRE ? : " ; A$
30  INPUT " QUEL EST VOTRE NO. S.S. ? ; " ; S$
40  IF A$ (1 ; 1) = " 1 " THEN GOTO 80
50  IF A$ (1 ; 1) = " 2 " THEN GOTO 100
60  PRINT " ERREUR DANS LE NUMERO DE S.S. "
70  GOTO 30
80  PRINT " BONJOUR MONSIEUR " ; A$
90  GOTO 20
100 PRINT " BONJOUR MADAME " ; A$
110 GOTO 20
```

En vous inspirant de l'exemple du chapitre 8, compléter ce programme pour le faire arrêter dès qu'un nom propre particulier apparaît (une chaîne vide par exemple).

Exercice 5

```
10   INPUT " VALEUR DE P ? " ; P
12   M = 0
15   PR " ∧ L1   ∧∧∧   L2  ∧∧∧∧  S "
16   PR " **** ∧∧ ****  ∧∧∧ *** "
20   FOR I = 1  TO  INT (P/2) + 1
30   L1 = I - 1   \  L2 = (P/2) - LI
40   S = L1 * L2
50   PR L1 ; TAB (7) ; L2 ; TAB (14) ; S
55   IF S < M THEN 90
60   M = S
70   M1 = L1
80   M2 = L2
90   NEXT I
100  PR
110  PR " SURFACE MAX : " ; M ; " ∧ POUR L1 ∧ =    " ; M1 ; " ∧ ET L2 = ∧ " ; M2
120  STOP
```

M1 et M2 mémorisent les valeurs de L1 et L2 correspondant à M. En cas d'égalité de deux surfaces S maximales, on constate que le test S < M donne une réponse NON et que donc on mémorise celle dont la valeur L1 est la plus grande (et donnera un champ... plus trapu). Vérifiez que ceci est vrai pour P multiple de 2, non multiple de 4.

Exercice 6

La longueur (inconnue) de la chaîne M$ est L = LEN (M$). Le dernier caractère a donc pour indice : L et l'avant-dernier : L − 1. Donc, la sous-chaîne comportant les deux derniers caractères d'un mot s'obtient en deux instructions :

```
L = LEN (M$)
S$ = M$ (L - 1 ; L)
```

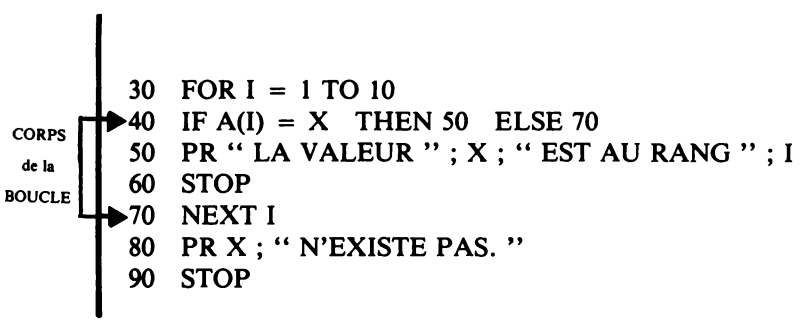
Exercice 7

```
5   DIM M$ (20)
10  INPUT " QUEL EST LE MOT ? : " ; M$
20  L = LEN (M$)
30  S$ = M$ (L – 11 ; L)
40  IF S$ = " OU " THEN PR " OUI " ELSE PR " NON "
50  STOP
```

S$ n'a pas besoin d'être dimensionné.

Le mot M$ doit avoir au moins 2 caractères, sinon il y a erreur ; mais on aurait pu vérifier que $2 \leqslant L \leqslant 20$.

Exercice 8




```
30  FOR I = 1 TO 10
40  IF A(I) = X  THEN 50  ELSE 70
50  PR " LA VALEUR " ; X ; " EST AU RANG " ; I
60  STOP
70  NEXT I
80  PR X ; " N'EXISTE PAS. "
90  STOP
```

Remarque très importante - En BASIC LX, on ne doit pas " sortir " d'une boucle par une instruction de type GOTO ; soit on arrête le programme à un certain rang, soit on sort (et on continue l'exécution en séquence) à la fin de la boucle. Tout branchement à une instruction extérieure au corps de la boucle risque d'entraîner (dans certains cas) une erreur.

Si l'organisation du programme l'exige, on peut sortir en utilisant l'instruction EXIT *n* qui interrompt l'exécution de la boucle et branche à l'instruction *n*. Le programme précédent s'écrirait :

```
30  FOR I = 1 TO 10
40  IF A(I) = X  THEN EXIT 80
50  NEXT I
60  PR X ; "  N'EXISTE PAS. "
70  STOP
80  PR " LA VALEUR  " ; X ; "  EST AU RANG " ; I
90  STOP
```

Exercice 9

- 1er mot de 1 à 20
- 2e mot de 21 à 40
- 3e mot de 41 à 60
- mot de rang X de $(X-1)\times 20+1$ à $X\pm 20$

Exercice 10

Il nous faut :

- connaître la longueur du mot X$:

$$L = LEN\ (X\$)$$

- vérifier qu'elle n'est pas supérieure à 20 (sinon, on rejettera le mot) ;
- si elle est inférieure à 20, on doit rajouter D = 20 − L étoiles ;
- si elle est égale à 20, il n'y a rien à faire.

Exercice 11

La chaîne U$ étant créée, un programme de recherche peut s'écrire :

```
10   E$ = " ******************** "
...
100  REM PROGRAMME DE RECHERCHE DE TOTO.
110  FOR I = 1 TO N/20
120  IF U$ ( (I - 1) * 20 + 1 ; I * 20) = " TOTO " + E$ (1 ; 16) THEN 130 ELSE 150
130  PR " TOTO EST TROUVE, IL EST  " ; I ; "  EME. "
140  STOP
150  NEXT I
160  PR " PAS DE TOTO A BORD ! "
170  STOP
```

Exercice 12

Si R3/R1 n'est jamais égal à x, il convient de choisir le jeu d'engrenages tel que la différence entre $Z = \frac{R3}{R1}$ et X soit minimale (en valeur absolue).

Nous avons déjà programmé la recherche d'un maximum, celle d'un minimum est fort semblable. Appelons E l'écart entre R3/R1 et X. Nous initialiserons E a $|1 - K|$ puisque le rapport 1 existe (R1 = R3) avec la combinaison R1 = R(1) et R3 = R(1) :

```
100  E = ABS (1 - X) \ R1 = R(1) \ R3 = R(1)
110  FOR I = 1 TO N
120  FOR J = 1 TO N
125  Z = ABS (R(J) / R(I) - X)
130  IF Z > = E THEN 150
140  E = Z \ R1 = R(I) \ R3 = R(J)
150  NEXT J
160  NEXT I
170  PR " LA MEILLEURE SOLUTION EST " ; R1 ; " ET " ; R3
180  GOTO 70
```

On remarquera que si Z = 0, il convient d'afficher toutes les solutions pour laisser l'utilisateur choisir.

Exercice 13

Par exemple, on cherchera à trouver une roue R2 de rayon proche de la moyenne des rayons des deux autres. Donc, on gardera la combinaison pour laquelle la valeur absolue de $R2-(\frac{R1+R3}{2})$ est minimale. C'est donc une recherche de minimum que vous devez savoir programmer maintenant. *(Attention à l'initialisation.)*

Chapitre XII

QUELQUES PROGRAMMES PRATIQUES EN BASIC

Allons maintenant plus avant dans l'emploi du Basic. Pour cela, nous allons développer une série de programmes destinés à apporter un éclairage nouveau sur tel ou tel aspect de ce langage. Présentés comme des exercices intellectuels ou comme des jeux, ces programmes pourront être facilement compris et prépareront le lecteur à l'étude complète du Basic et de ses possibilités.

1. EXTRACTION DE RACINES CARREES ET CUBIQUES

A quoi bon rédiger un programme pour extraire une racine carrée puisque l'instruction correspondante existe déjà en Basic ? Oui, mais il n'en va plus de même pour les racines cubiques... Quand bien même, d'ailleurs : l'étude de l'algorithme utilisé est intéressante et montre que la machine informatique procède d'une façon différente de celle que nous pratiquons. De plus, le lecteur pourra se familiariser avec la manière de bâtir un ordinogramme (ou un « organigramme ») de même qu'avec quelques instructions dont le rôle et l'effet sont ici mis en évidence.

1.1. L'algorithme de base pour racines carrées

Les méthodes classiques d'extraction de racines carrées ne sont guère adaptées à l'usage d'un ordinateur. On préfère souvent procéder par itérations. Ainsi, soit a la valeur dont on veut extraire la racine ; on fera successivement :

$$x_1 = \frac{1}{2}\left(x_0 + \frac{a}{x_0}\right)$$

$$x_2 = \frac{1}{2}\left(x_1 + \frac{a}{x_1}\right)$$

$$x_3 = \frac{1}{2}\left(x_2 + \frac{a}{x_2}\right)$$

...etc., chaque valeur successive de x donnant une racine plus précise, et ce jusqu'à :

$$x_n = \frac{1}{2}\left(x_{n-1} + \frac{a}{x_{n-1}}\right)$$

On commence par estimer x_0, puis les valeurs successives de x qu'on peut appeler x_i pour $i = 1, 2, 3, \ldots, n$ et qui sont des approximations de la racine carrée de a. Chaque calcul successif de x fournit une meilleure approximation, jusqu'à x_n qui est la valeur $\sqrt{x}$ donnée avec la précision apportée par le degré n.

Afin d'exploiter ce processus, on estime d'abord $\sqrt{a}$ et on l'appelle x_0. Puis, on divise a par x_0 et on prend la moyenne de x_0, ce qui mène à une approximation meilleure, x_1. L'algorithme général peut donc s'écrire :

$$x_n = \frac{1}{2}\left(x_{n-1} + \frac{a}{x_{n-1}}\right)$$

La procédure converge toujours vers la meilleure approximation, indépendamment du choix initial de x_0 mais à la condition que x_0 et a soient positifs. A quel moment cependant faut-il la faire stopper ? On se contentera, ici, de définir une erreur maximale admissible e telle que :

$$|x_n - x_{n-1}| < e$$

expression où les barres verticales indiquent la valeur absolue, de façon habituelle.

La méthode n'est cependant pas sans inconvénients ; ainsi, si a est un carré parfait et si l'on ne choisit pas d'emblée $x_0 = \sqrt{a}$, on n'aboutira jamais à la valeur exacte de $\sqrt{a}$: chaque opération ne fait qu'approcher cette valeur. Pourtant, si le nombre d'opérations est suffisant et si l'écart e toléré est faible, on finit

par obtenir le résultat escompté à la condition d'éliminer des décimales gênantes. Par exemple, soit à calculer $\sqrt{4}$ en prenant, au départ, $x_0 = 1$ et en ne retenant que deux chiffres après la virgule :

$x_0 = 1{,}00$

$$x_1 = \frac{1}{2}\left(1{,}00 + \frac{4{,}00}{1{,}00}\right) = 2{,}50$$

$$x_2 = \frac{1}{2}\left(2{,}50 + \frac{4{,}00}{2{,}50}\right) = 2{,}05$$

$$x_3 = \frac{1}{2}\left(2{,}05 + \frac{4{,}00}{2{,}05}\right) = 2{,}00$$

Apparemment, on a bien obtenu le bon résultat mais si l'on avait retenu les décimales suivantes, on aurait constaté que $x_3 = 2{,}0006\ldots$, erreur que x_4 ou $x_5\ldots$ auraient encore réduite.

Prenons un autre exemple : soit à calculer la racine de 5476, qui est le carré parfait de 74. Adoptons 40 pour x_0. On aura :

$x_0 = 40$

$$x_1 = \frac{1}{2}\left(40 + \frac{5476}{40}\right) = 88{,}45$$

$$x_2 = \frac{1}{2}\left(88{,}45 + \frac{5476}{88{,}45}\right) = 75{,}18034$$

$$x_3 = \frac{1}{2}\left(75{,}18034 + \frac{5476}{75{,}18034}\right) = 74{,}009265$$

$$x_4 = \frac{1}{2}\left(74{,}009265 + \frac{5476}{74{,}009265}\right) = 74{,}00000$$

Le résultat est donc excellent après quatre itérations, ce qui témoigne que le plus souvent, l'erreur éventuelle n'aura aucune portée. Par conséquent, on peut sans crainte élaborer un ordinogramme (fig. 12.1) et le programme correspondant à cet algorithme.

Pour cela, on va appeler *a* la valeur dont on veut extraire la racine et *e* l'erreur définie ci-dessus et qu'on fixe, ici, à 0,0001 (mais rien ne s'oppose à ce que vous adoptiez d'autres valeurs, par exemple : 0,00001 ou 0,001...). On va se proposer d'extraire la racine carrée de 625, comme l'indique la ligne 160 du programme mais à nouveau, vous pouvez sélectionner tout autre nombre.

La ligne 30 appelle donc *a* (soit 625) et *e* (soit 0,0001) et illustre le rôle de l'instruction READ, ordre de lire à la suite les données présentées par l'instruction DATA (ligne 160).

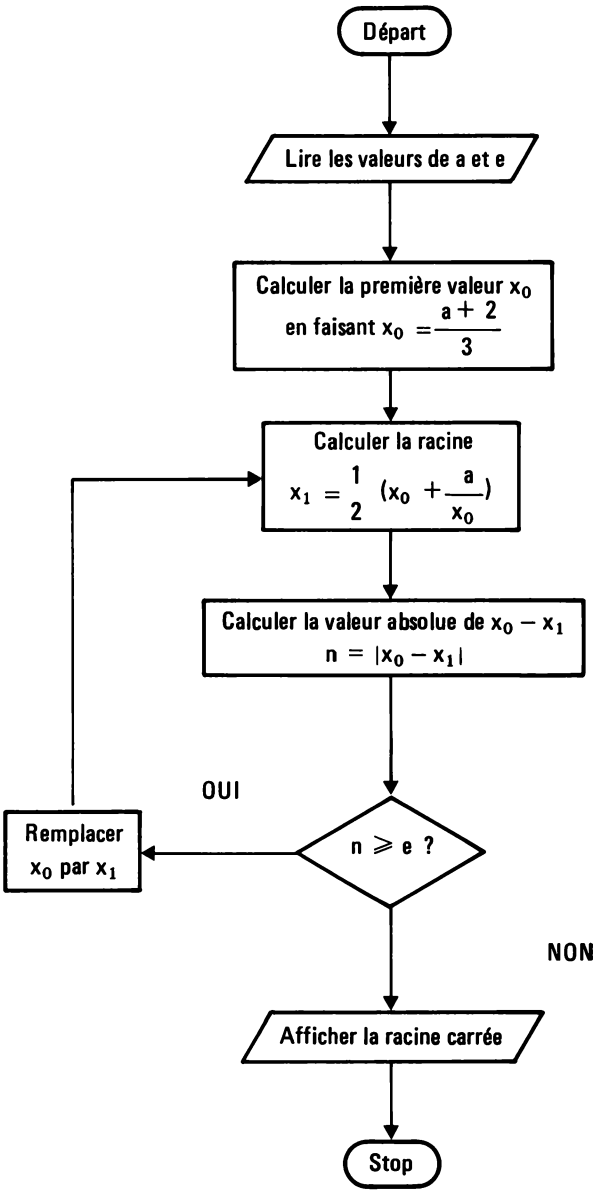


Fig. 12.1 — Organigramme de l'extraction d'une racine carrée d'un nombre.

```
10 REM                    EXTRACTION DE RACINES CARREES
20 REM                    PAR APPROXIMATIONS SUCCESSIVES
30 READ A;E
40 LET X0 = (A+2)/3
50 LET X1 = 0.5*(X0+A/X0)
60 REM CALCUL DE L'ERREUR
70 LET N = X1-X0
80 IF N >= 0 THEN 100
90 LET N = -N
100 IF N < E THEN 140
110 REM APPROXIMATION SUIVANTE
120 LET X0 = X1
130 GOTO 50
140 PRINT "LA RACINE CARREE DE ";A;" EST ";X1
150 END
160 DATA 625 ; 0.0001



LA RACINE CARREE DE 625 EST 25
```

La ligne 40 sert à calculer la valeur initiale de x_0 et la ligne 50, la première approximation. Puis on examine ce que vaut $x_1 - x_0$ pour savoir si cette différence est plus petite ou non que e que l'on s'est fixée. Le résultat branchera sur une nouvelle itération (ligne 130 commandant le GOTO à la ligne 50), ou commandera l'affichage du résultat, 25 en toute rigueur :

```
RUN

LA RACINE CARREE DE 625 EST 25
```

1.2. Avec l'instruction ABS

On peut cependant simplifier un tel programme en recourant à une instruction qui fournit la valeur absolue d'une grandeur. Elle est notée ABS. Si l'on écrit ABS (X1 – X0), on obtiendra immédiatement la valeur absolue de cette différence et l'on pourra remplacer plusieurs lignes du programme précédent par une seule, ce qui donne :

```
10 REM                EXTRACTION DE RACINES CARREES
20 REM                      AVEC "ABS"
30 READ A;E
40 LET X0 = (A+2)/3
50 LET X1 = .5*(X0+A/X0)
60 REM UTILISONS MAINTENANT ABS
100 IF ABS (X1-X0) < E THEN 140
120 LET X0 = X1
130 GOTO 50
140 PRINT "LA RACINE CARREE DE ";A;" EST ";X1
150 END
160 DATA 625 ; .001
```

En lançant le programme, on obtient :

```
RUN

LA RACINE CARREE DE 625 EST 25
```

1.3. Extraction de la racine cubique

Une méthode semblable permet l'extraction de racines cubiques. Ainsi, si l'on dispose d'une valeur approximative y_0 de la racine, une meilleure approximation y_1 sera fournie par :

$$y_1 = \frac{1}{3}\left(\frac{x}{y_0^2} + 2y_0\right),$$

où x est le nombre dont on veut extraire la racine cubique. En généralisant, on peut écrire :

$$y_{i+1} = \frac{1}{3}\left(\frac{x}{y_i^2} + 2y_i\right),$$

Comme précédemment, on procède par itérations jusqu'à ce que l'erreur admissible e soit inférieure à la valeur E qu'on s'est fixée, par exemple, à 1/3, ce qui s'écrit :

$$|y_{i+1} - y_i| < E$$

Dans ce programme, on va fournir la première approximation y_1 de la racine. Cette façon de faire pourra, comme dans les deux programmes d'extrac-

tion de racines carrées, paraître astreignante ; aussi, une autre méthode consistant à demander à l'opérateur d'introduire ses valeurs (instruction INPUT) peut être envisagée. On réservera cette nouvelle version à l'exercice proposé.

Ici, on va chercher la racine cubique de 3375 avec $e = 0{,}333333333333$. L'ordinogramme correspondant est donné figure 12. 2.

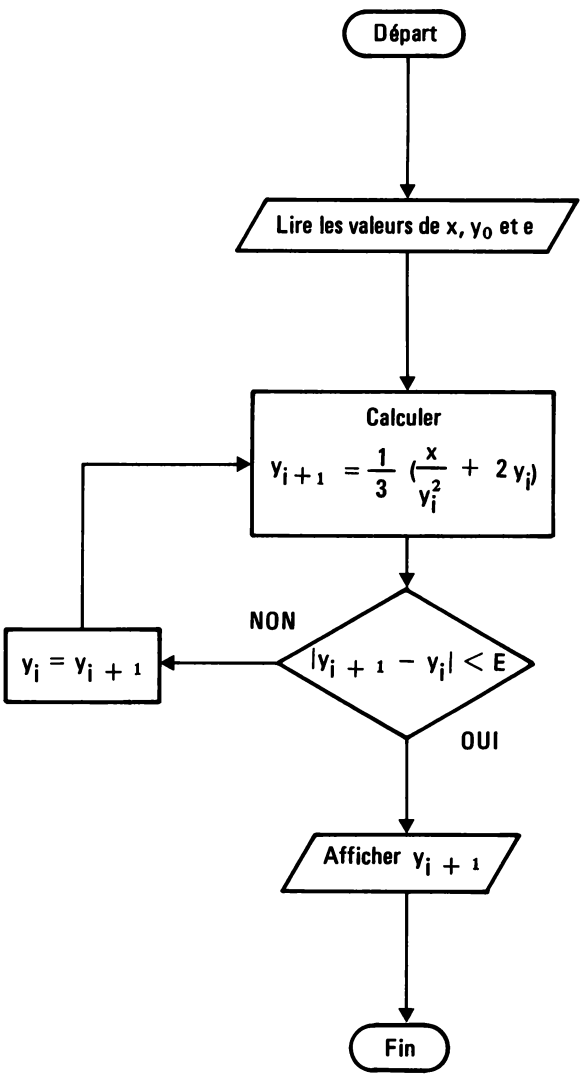


Fig. 12.2 — Organigramme pour l'extraction d'une racine cubique.

```
100 REM                 CALCUL DE LA RACINE CUBIQUE
110 REM                 PAR ITERATIONS
120 READ X;Y;E
130 LET Y1 = .333333333333*(X/Y^2+2*Y)
140 IF ABS (Y1-Y)   E THEN 170
150 LET Y = Y1
160 GOTO 130
170 PRINT "LA RACINE CUBIQUE DE ";X;" EST ";Y1
180 END
190 DATA 3375 ; 10 ; .0001



LA RACINE CUBIQUE DE 3375 EST 15
```

En lançant ce programme, on obtient :

RUN

LA RACINE CUBIQUE DE 3375 EST 15

Exercice :

Ces programmes d'extraction de racines sont figés parce qu'on a proposé, une fois pour toutes, les valeurs sur lesquelles on travaille.

Transposez ces programmes en introduisant des demandes de données « fraîches » à l'opérateur, et en précisant de quelle donnée il s'agit, par exemple :

PRINT " INTRODUISEZ LA VALEUR X "
PRINT " DONT ON VA CALCULER
PRINT " LA RACINE "
INPUT X

puis

PRINT " QUELLE PRECISION E "
PRINT " SOUHAITEZ-VOUS ? "
INPUT E
et pour une racine cubique :
PRINT " PROPOSEZ UNE PREMIERE "
PRINT " VALEUR Y : "
INPUT Y
A chaque RUN, vous pourrez effectuer un nouveau calcul.

2. CHAINES DE CARACTERES : LES 7 JOURS DE LA SEMAINE

La manipulation de chaînes de caractères surprend souvent les débutants. Pour nous familiariser avec elle, voici un petit programme sans ambition autre que celle de bien faire comprendre ce qu'est réellement une chaîne.

Nombreuses sont les versions du Basic qui procurent le moyen de traiter des *chaînes de caractères ;* pas toutes cependant. Certains Basic appelleront « sous-chaîne » ce que d'autres appellent « chaînes », traiteront d'ensembles chaînés de longueurs fixes, etc., ce qui prouve qu'il faudra toujours se référer à la notice du Basic utilisé.

Soit une suite de caractères, tels que : LUNDI. Et puisqu'il y a sept jours dans la semaine, on aura sept chaînes : lundi, mardi, mercredi, jeudi, vendredi, samedi et dimanche, ce qui n'apprendra certainement rien au lecteur...

Par contre, et pour indiquer en Basic qu'il s'agit d'une chaîne de caractères alphabétiques, on utilise le symbole représentant le mot « dollars », qui s'écrit $. Ainsi noterons-nous J$, une chaîne représentant un jour de la semaine.

Proposons-nous donc de rédiger un programme qui imprimera ou affichera les 7 jours de la semaine dans l'ordre, puis dans l'ordre inverse. Voici ce que cela pourrait donner :

```
10 PRINT "VOICI LES 7 JOURS DE LA SEMAINE"
20 PRINT "1.DANS L'ORDRE : "
30 DIM J$(56)
40 FOR I = 1 TO 56 STEP 8
50     READ J$(I;I+7)
60 NEXT I
70 FOR I = 1 TO 56 STEP 8
80     PRINT J$(I;I+7)
90 NEXT I
100 PRINT
110 PRINT "2.DANS L'ORDRE INVERSE :
120 FOR I = 56 TO 1 STEP -8
130    PRINT J$(I-7;I)
140 NEXT I
145 END
150 DATA "LUNDI" ; "MARDI" ; "MERCREDI" ; "JEUDI"
160 DATA "VENDREDI" ; "SAMEDI" ; "DIMANCHE"
```

Dans ce programme, on voit qu'on a utilisé un pas négatif (ligne 120) pour l'ordre inverse. Voilà ce que donne son lancement :

```
VOICI LES 7 JOURS DE LA SEMAINE
1.DANS L'ORDRE :
LUNDI
MARDI
MERCREDI
JEUDI
VENDREDI
SAMEDI
DIMANCHE

2.DANS L'ORDRE INVERSE :
DIMANCHE
SAMEDI
VENDREDI
JEUDI
MERCREDI
MARDI
LUNDI
```

Exercices :

1) Rédigez un programme semblable avec les 12 mois de l'année.

2) Commandez l'impression du premier mois de chaque trimestre dans ce dernier programme.

3. ALGORITHMES DE TRI NUMERIQUE ET ALPHABETIQUE

Le problème du classement par ordre numérique se pose fréquemment. Comment procède-t-on ? C'est ce qu'on va voir ici à l'aide d'un petit exercice de rangement de nombre par ordre croissant.

De nombreux algorithmes de classement existent ; en général, ils procèdent par comparaisons et permutations.

Comment permute-t-on deux valeurs ? Si l'on tient compte que la première valeur est logée en mémoire à A et la seconde à B, il faut utiliser une troisième référence mémoire qui sera, par exemple, M. On fait passer le contenu de A dans M, celui de B dans A et enfin celui de M dans B. Ainsi, M n'a fait que servir de relais intermédiaire.

Ainsi, soit à trier et mettre dans l'ordre croissant les chiffres 5, 2, 6, 4 et 3. On compare 5 à 2 et, puisque 5 est plus grand, on permute ces deux valeurs de la façon indiquée. La nouvelle liste est alors : 2, 5, 6, 4, 3.

On continue ensuite, en comparant 5 au nombre suivant, 6 : ils sont dans le bon ordre et par conséquent, on n'opère aucune permutation.

Mais en comparant ensuite 6 à 4, on s'aperçoit qu'il faut les permuter, ce que l'on fait aussitôt, et la liste devient 2, 5, 4, 6, 3. Bien sûr, quelque chose « cloche » mais la machine n'en a garde pour l'instant et continue, en comparant 6 à 3 et en exécutant la permutation imposée, ce qui donne pour la liste 2, 5, 4, 3, 6.

Ainsi se termine ce premier tour, qu'on appelle une première passe. La seconde va alors commencer selon la même technique, avec les étapes suivantes :

A — Au départ, on a donc 2, 5, 4, 3, 6.
B — Première comparaison : 2,5. Tout va bien.
C — 2e comparaison : 5 et 4, on permute. On obtient alors 2, 4, 5.
D — 3e comparaison : 5 et 3, on permute et l'on obtient 2, 4, 3, 5.
E — 4e comparaison : 5 et 6, tout va bien.

La série à la fin de la seconde passe est 2, 4, 3, 5, 6. La troisième passe donnera 2, 3, 4, 5, 6 et la quatrième témoignera, parce qu'il n'y a plus de permutation, que le classement est terminé.

On démontre que pour trier N « objets », il faut au maximum N – 1 passes. Comment rédiger le programme correspondant ? Avant de le présenter, nous dresserons la liste et la signification des lettres qui seront utilisées :

• P est le nombre maximal d'objets à classer, de 50 dans notre programme (car on se réserve une marge importante),

• X le nombre réel d'« objets » à classer, donc 5 ici. Par conséquent, le nombre maximal de passes sera de 5 – 1 = 4.

• C'est T qui, pratiquement, contrôle le nombre de passes puisqu'il va de 1 à X.

• Par contre, R agit comme un pointeur et se déplace le long de la liste à chaque passe.

• Enfin, N est un témoin, un indicateur qui reste à zéro s'il n'y a eu aucune permutation mais passe à 1 dès qu'une permutation est exécutée. On saura, en le consultant à chaque fin de passe, s'il y a eu permutations : en l'absence de celles-ci, le classement est terminé (donc, lorsque N = 0).

```
10 DIM P(50)
20 READ X
30 FOR T = 1 TO X
40         READ P(T)
50 NEXT T
60 PRINT "LISTE INITIALE :
65 FOR T = 1 TO X
70         PRINT P(T)
75 NEXT T
80 FOR T = 1 TO X   1
90         N = 0
100        FOR R = 1 TO X - T
110            IF P(R+1) >= P(R) THEN 160
120            A = P(R)
130            P(R) = P(R+1)
140            P(R+1) = A
150            N = 1
160        NEXT R
170        IF N = 0 THEN EXIT 190
180 NEXT T
190 PRINT
200 PRINT "ET VOICI DANS L'ORDRE :
210 FOR T = 1 TO  X
220        PRINT P(T)
230 NEXT T
235 END
240 DATA 5
250 DATA 5;2;6;4;3
```

La première ligne du programme déclare la dimension du tableau P, numérique, susceptible de contenir 50 valeurs, mais X précisera ensuite qu'il n'y en a réellement que 5. On commence par imprimer ces valeurs dans le désordre, ce qui fait l'objet des instructions jusqu'à la ligne 75

Puis, on s'occupe du nombre de passes, qu'indique T, et qui va de 1 à 5 – 1 = 4 (ligne 80). L'indicateur noté N témoigne, lorsqu'il se met à 1, qu'il y a eu une permutation ; au départ, on lui attribue la valeur zéro (ligne 90). Le pointeur R, lui, va d'abord évoluer entre R = 1 et R = 5 – 1 = 4.

Commence alors le processus de comparaison (ligne 110) et de permutations (lignes 120 à 140), où A sert de relais, comme on l'a vu tout au début de l'étude de ce programme. S'il y a eu permutation, on met N à 1 (ligne 150), et l'on passe au R suivant, ligne 160, à laquelle on aurait directement abouti s'il n'y avait pas eu de permutation (le « THEN 160 » de la ligne 110).

On voit (ligne 170) que s'il n'y a pas eu de permutation, N = 0 et l'on passe à la fin de la séquence puisque le classement est terminé. Mais s'il y a eu permutation, on engage la passe suivante (« NEXT T » de la ligne 180).

A partir de la ligne 190 et en commençant par sauter une ligne à l'impression, on édite le tableau dûment classé. La première donnée, 5 (de la ligne 240) se rapporte à X, celles de la ligne 250 sont les nombres à classer.

Remarquons encore la mesure d'économie prise à la ligne 100 : à chaque passe, on réduit de 1 le nombre de comparaisons puisqu'il est certain qu'à chaque tour de piste, on chasse le nombre le plus fort au bout du tableau !

Par conséquent et si on lance ce programme, on obtient :

```
LISTE INITIALE :
5
2
6
4
3

ET VOICI DANS L'ORDRE :
2
3
4
5
6
```

On remarquera la façon de rédiger ce programme, avec des lignes en retrait : c'est pour rendre plus évidentes les diverses boucles « FOR » et faciliter sa lecture et sa compréhension.

Tel était bien le résultat à obtenir.

Exercice :

Dans ce programme, on a placé des ordres sans préciser l'instruction ; par exemple, dans les lignes 90 et 150. En effet, certains Basic les acceptent tels. Mais d'autres les rejetteront. Quelle instruction d'affectation faudrait-il alors placer sur ces lignes ? Ré-écrivez-les.

4. LE TRACE DE SINUS x

Les fonctions graphiques sont intéressantes à plus d'un titre. On peut les aborder à l'aide d'un court programme qui commande le tracé de la courbe y = sin x pour quelques valeurs de x. Cela va nous permettre d'examiner le rôle de la fonction « tabulation » et de la fonction « entier » *(« integer »)*. Après quoi, vous pourrez vous lancer dans des programmes plus compliqués de dessin...

4.1. Calcul des valeurs de sin x

La fonction mathématique *sinus x* peut être étudiée en écrivant un programme qui, via une boucle « FOR » introduisant des arguments successifs, permet de dresser une table des valeurs correspondantes.

En Basic, les fonctions trigonométriques sont en radians et non en degrés. Rappelons que 2 π radians = 360° ; 6,28 radians (soit 2 π) valent donc 360° et un radian vaut environ 57°.

Au lieu de 6,28 comme argument maximal, prenons 6 pour simplifier les choses et étudier :

$$y = \sin x$$

de 0 à 6, par pas de 1/2, ce qui donnera 13 points de repère. On va ainsi pouvoir examiner le fonctionnement de la boucle FOR X = 0 TO 6 STEP .5 qui traduit cette instruction.

Le programme est ainsi le suivant :

```
10 PRINT " X" ; TAB(7) ; "Y = SIN(X)"
20 FOR X = 0 TO 6 STEP .5
30      LET Y = SIN(X)
40      PRINT X ; TAB(7) ; Y
50 NEXT X
60 END
```

La première instruction (ligne 10) définit les têtes de colonnes.

La ligne 20 sert à calculer les arguments successifs, c'est-à-dire les valeurs attribuées à X (en radians), et la ligne 30 à calculer $y = \sin x$. Puis, on affiche ou imprime x et y, toujours en deux colonnes séparées de 7 signes et l'on passe à la valeur suivante de x, donc à la valeur précédente accrue de 0,5. Voici ce que donne le lancement de ce programme :

```
X       Y = SIN(X)

0        0
.5       .479425539168
1        .8414709928
1.5      .99749500148
2        .90929743345
2.5      .598472145286
3        .14112000806 3
3.5      -.350783227907
4        -.756802498154
4.5      -.97753012877
5        -.95892428404
5.5      -.705540327716
6        -.279415498283
```

Comme c'est souvent le cas, de telles listes de nombres ne sont pas très parlantes et l'on en revient toujours à l'idée qu'une bonne courbe vaut mieux que bien des discours. Mais avant de la tracer, simplifions le programme maintenant qu'on en a compris le principe, en remplaçant les lignes 30 et 40 par une seule, de la façon suivante (on obtient rigoureusement le même résultat) :

```
10 PRINT " X" ; TAB(7) ; "Y = SIN(X)"
15 PRINT
20 FOR X = 0 TO 6 STEP .5
30 PRINT X ; TAB(7) ; SIN(X)
40 NEXT X
50 END
```

Ici, en effet, le calcul de sin x est donné dans la même ligne qui commande son impression.

4.2. Tracé avec l'ordre TAB

Passons au tracé de cette courbe, en respectant les points définis par les arguments successifs de x (fig. 3). Pour obtenir un meilleur tracé, on va étudier cette courbe de 0 à 6,3 radians (donc, un peu plus de 360°) par pas de 0,107. Voyons le programme nécessaire, qui sera commenté aussitôt après :

```
10 FOR K = -1 TO 1.1 STEP .2
20     PRINT TAB (9+30*(K+1)) ; INT (K*100)/100 ;
30 NEXT K
40 PRINT
50 FOR X = 0 TO 6.3 STEP .107
60     PRINTX ; TAB (10+30*(SIN(X)+1)) ; "*"
70 NEXT X
80 END
```

Les trois premières lignes définissent les têtes de colonnes, c'est-à-dire les valeurs des sinus de -1 à $+1$. On a indiqué 1,1 mais cette valeur ne peut être atteinte (et affichée) puisque le pas est de 0,2. L'ordre de *tabulation* (TAB) indique l'espacement entre points successifs, selon une échelle linéaire ; la première colonne (-1) commence ainsi à $9 + 30(-1+1) = 9$ espaces à partir de la marge ; la dernière sera à $9 + 30(+1-1) = 69$ espaces à partir de la marge. On dispose ainsi de $69 - 9 = 60$ colonnes.

Mais à quoi correspond alors INT (K * 100) / 100 ? On a voulu limiter le nombre de décimales après la virgule et pour cela, on utilise l'ordre INT, (pour « *integer* », soit « entier »), qui fonctionne de la façon suivante. Soit x = 3,1416 :

$$x = 3{,}1416$$
$$100 * x = 314{,}16$$

la partie entière de 100 x est alors :

$$\text{INT}(100 * x) = 314$$

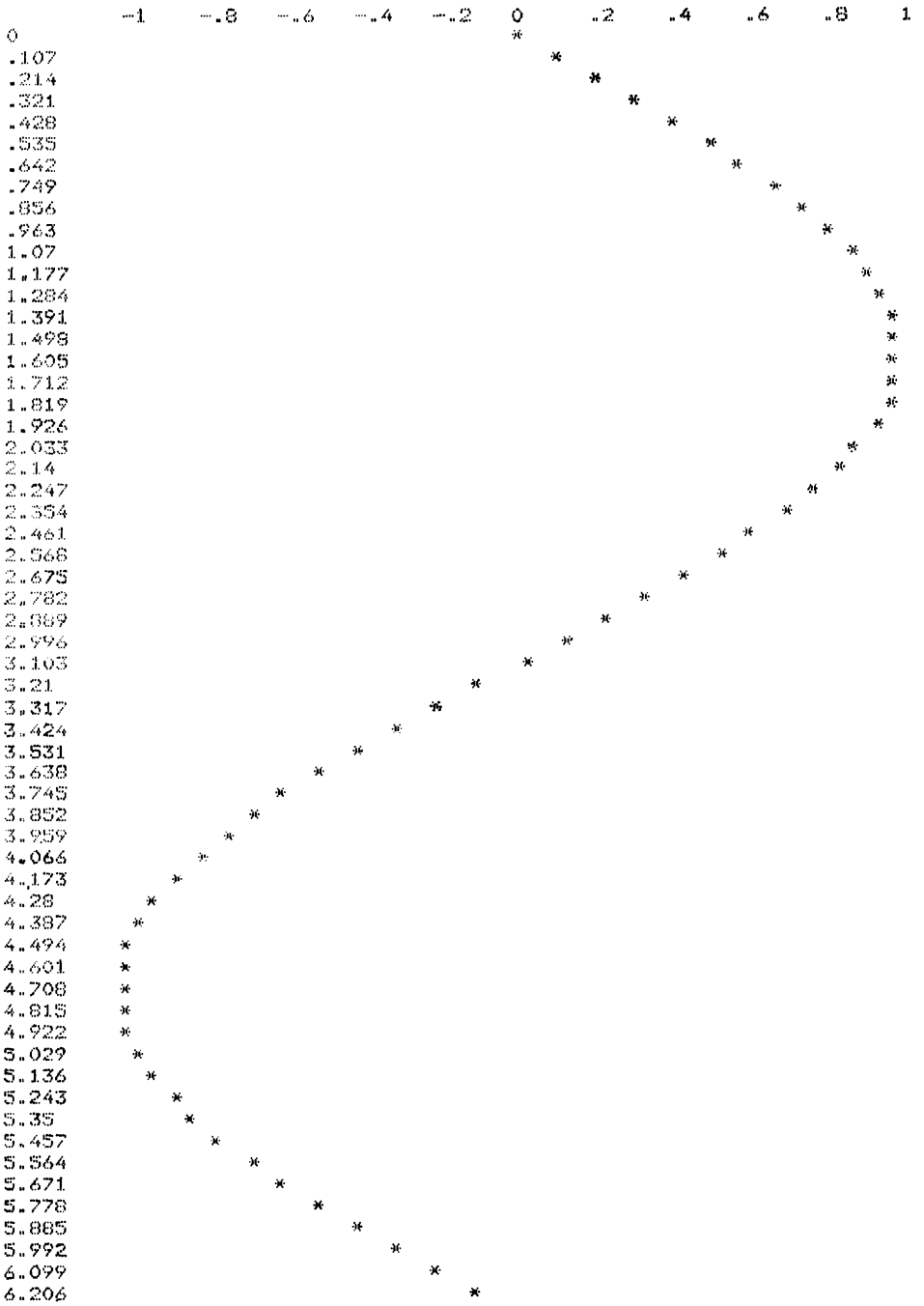


Fig. 12-3 — Le tracé de sinus x. La courbe est quelque peu aplatie à ses maximums en raison des possibilités de l'imprimante utilisée par l'auteur

et

$$\text{INT } (100 * x) / 100 = 3,14$$

On a ainsi ramené à deux le nombre de décimales, ce qui pourra être utile avec certaines machines ou certains Basic.

Après quoi, on calcule les arguments successifs et l'on commande l'impression d'un astérisque (*) à l'emplacement calculé : telle est la fonction de la ligne 60 dont le rôle est double :

1) afficher l'argument *x*.

2) afficher l'astérisque à un emplacement fonction de sinux *x*, ce qu'indique à nouveau la tabulation (TAB).

On a ici utilisé SIN(X) + 1 parce que la tabulation ne fonctionne pas sur des valeurs négatives ! Elle ira (pour elle) de 0 à 2 et, pour couvrir les 60 colonnes, on multiplie le tout par 30. Enfin, afin de laisser de la place pour l'affichage de *x*, on décale le tout de + 10. La courbe ira donc des colonnes 10 à 70 et se situe bien en-dessous des têtes de colonnes affichées.

Si votre terminal ne dispose pas du nombre suffisant de colonnes, vous prendrez 15 au lieu de 30 dans les deux tabulations. Et si vous préférez autre chose que des astérisques, ne vous en privez pas...

4.3. Si votre Basic ne dispose pas de l'ordre TAB

Certains Basic ne comportent pas l'ordre TAB, ou encore l'acceptent mais avec des constantes obligatoires. Dans ce cas, on est contraint de le remplacer par une série d'instructions aboutissant au même effet. Ainsi, au lieu d'écrire :

```
100  PRINT TAB (8) ; " *"
```

on proposera :

```
100  FOR  B = 0  TO  7
110  PRINT "  " ;
120  NEXT B
130  PRINT " * "
```

La tabulation de 8 a été ici remplacée par 8 − 1 = 7, car en réalité, elle commence par la position zéro. Le résultat obtenu sera identique au prix, bien sûr, d'un allongement du programme.

Exercice :

Rédiger un programme commandant le tracé de la courbe :

$$y = \cos x$$

5. LES CARRES MAGIQUES

Un carré magique, c'est un carré divisé comme un damier en cellules dans lesquelles on a inscrit des nombres. La somme de ces nombres selon une droite, horizontale ou verticale, ou selon une diagonale est toujours la même. Voici un algorithme qui va vous permettre d'en construire aisément et qui aura l'avantage de vous introduire dans le domaine des tableaux à deux dimensions.

Pour terminer cette série d'exercices en Basic, on va traiter d'un jeu mathématique, celui des carrés magiques. La figure 12-4 en montre un, de trois sur trois : si l'on additionne les chiffres contenus dans les cellules selon des axes verticaux, horizontaux ou en diagonale, on constate que le total fait toujours 15.

8	1	6
3	5	7
4	9	2

Fig. 12.4 — Un carré magique complet de 3 sur 3. La somme des rangées, colonnes ou diagonales est 15.

La conception de tels carrés a pendant des siècles constitué une récréation mathématique. Tenter d'en écrire un réclame pas mal de patience si l'on ne dispose pas du fil conducteur. Essayez, par exemple, d'en composer un autre, de 4 sur 4 ou 5 sur 5, en utilisant des nombres de deux chiffres...

Le « nombre magique », celui qui correspond au total qu'on trouve dans toutes les directions, est donné par la formule $n\ (n^2+1)\ /\ 2$, dans laquelle n est la dimension du carré. Et c'est un mathématicien jésuite, De La Loubère qui, au 17e siècle, a développé l'algorithme qu'on va exploiter avec la machine informatique.

Le processus est le suivant, en supposant que le nombre de cellules d'un côté est *impair* :

• On place un 1 au centre de la rangée supérieure.

• On imagine une cellule placée en diagonale, en haut et à droite de celle qu'on vient de remplir et dans laquelle on loge la valeur suivante immédiatement supérieure, donc 2. C'est ce que montre la figure 12-5.

• Cela fait et parce que la cellule nouvelle en diagonale est hors du carré, on ramène sa valeur dans la cellule opposée la plus lointaine dans la même colonne, ici, ou la même rangée si on a marqué une rangée.

• Toujours selon la même règle, on « invente » une cellule dans laquelle on place 3, valeur ramenée au bout de la rangée correspondante, comme le montre la même figure 12-5.

• Maintenant que trois valeurs ont été placées dans ce carré de 3 × 3, on change de série de cellules et on descend d'une cellule par rapport à la dernière. Tout va bien puisque celle-ci se trouve dans le carré : on y écrit la valeur suivante, 4.

• Selon le même principe de la diagonale, on va écrire 5, et 6.

• Puis, la deuxième série de chiffres étant inscrite, on descend d'une cellule et on écrit 7.

• Le 8 viendra sur la diagonale et sera rejeté au bout de la rangée.

• On procèdera de même pour le 9 : le carré est désormais complet !

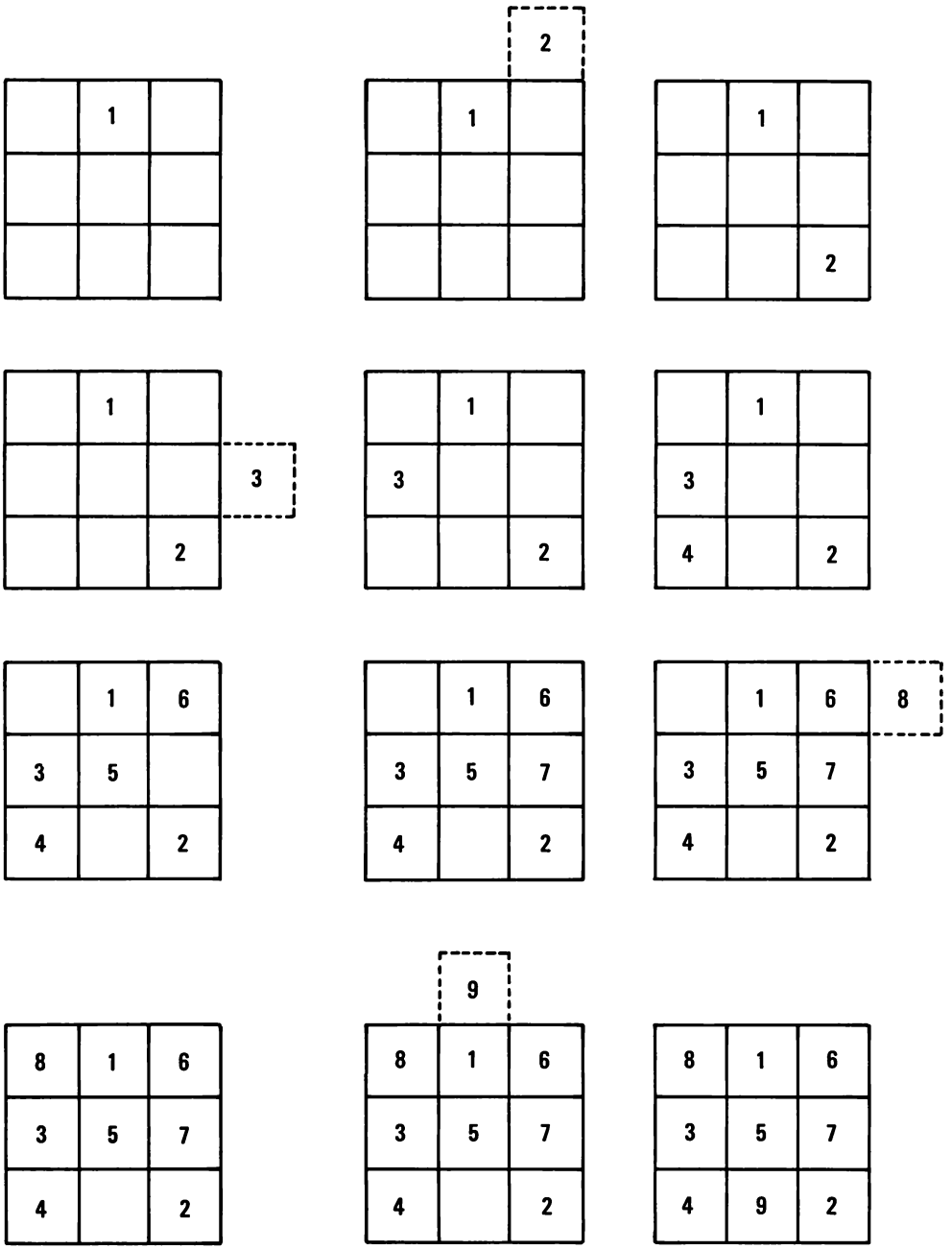


Fig. 12.5 — Décomposition du remplissage du carré magique, commentée dans le texte.

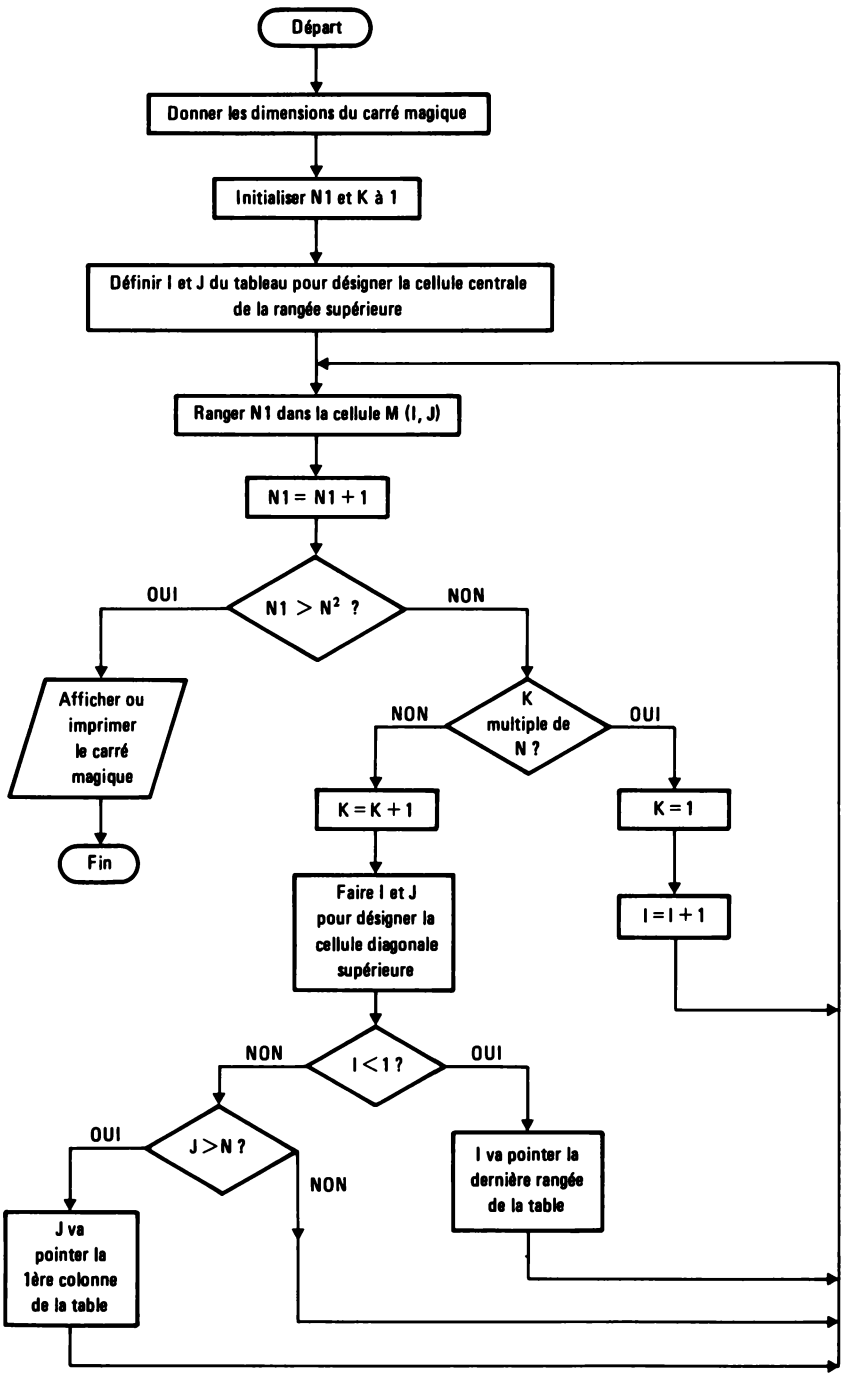

Départ
Donner les dimensions du carré magique
Initialiser N1 et K à 1
Définir I et J du tableau pour désigner la cellule centrale de la rangée supérieure
Ranger N1 dans la cellule M (I, J)
N1 = N1 + 1
N1 > N² ?
OUI
NON
Afficher ou imprimer le carré magique
Fin
K multiple de N ?
NON
OUI
K = K + 1
K = 1
Faire I et J pour désigner la cellule diagonale supérieure
I = I + 1
I < 1 ?
NON
OUI
J > N ?
OUI
NON
J va pointer la 1ère colonne de la table
I va pointer la dernière rangée de la table

Fig. 12.6 — Ordinogramme de l'établissement d'un carré magique.

Exercice :

Essayez de remplir un carré magique de 5 par 5 selon ce principe.

Vérifiez votre solution en la comparant à celle que fournira l'ordinateur, puisque c'est le problème qu'on va lui demander de résoudre.

L'algorithme est défini par l'ordinogramme de la figure 12-6. On commence par fixer le nombre N de cellules du côté du carré magique ; le compte de ces cellules est N1 que l'on initialise à 1 au début. On en fait autant avec un pointeur K, qui servira à déterminer les multiples de N (afin qu'on puisse changer de cellule et passer à celle immédiatement en-dessous).

Le carré magique lui-même est un tableau en deux dimensions, défini par des rangées (I) et des colonnes (J) de telle façon que chaque cellule puisse se voir attribuer un code de référence. Ainsi, M(I,J) avec pour I et J des valeurs comprises entre 1 et N indique une cellule précise ; c'est ce que montre la figure 12-7 pour un carré de 3 par 3. Par conséquent, et pour démarrer par la cellule centrale de la rangée du haut, il faut initialiser I à 1 et J à (N + 1) / 2. Dans cette cellule, on écrira la valeur N1.

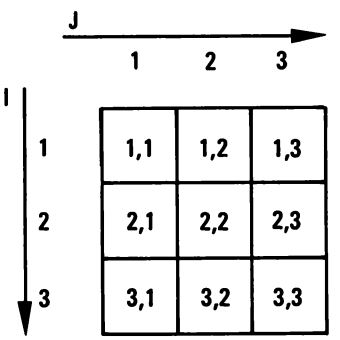


Fig. 12.7 — Les deux dimensions du tableau sont définies par les valeurs de I et de J qui vont, ici, de 1 à 3. Chaque case est donc précisée par (I,J).

Puis, on passe à la cellule suivante, N1 + 1 et on vérifie qu'on n'a pas dépassé le compte total des cellules qui est N^2. Si oui, on procède à l'impression du carré. Si non, on demande si K est un multiple de N.

```
10 REM                  TRACE D'UN CARRE MAGIQUE
20 REM                          DE N PAR N
25 DIM M(5;5)
30 LET N = 5
40 LET K = 1
50 LET N1 = 1
60 LET I = 1
70 LET J = (N+1)/2
80 LET M(I;J) = N1
90 LET N1 = N1 + 1
100 IF N1 > N^2 THEN 240
110 IF K < N THEN 150
120 LET K = 1
130 LET I = I + 1
140 GOTO 80
150 LET K = K + 1
160 LET I = I - 1
170 LET J = J + 1
180 IF I <> 0 THEN 210
190 LET I = N
200 GOTO 80
210 IF J <= N THEN 80
220 LET J = 1
230 GOTO 80
240 PRINT "CARRE MAGIQUE DE ";N;" PAR ";N
250 PRINT
260 FOR I = 1 TO N
270       FOR J = 1 TO N
280           PRINT TAB(J*4); M(I;J);
290       NEXT J
300       PRINT
310       PRINT
370 NEXT I
380 END
```

Le pointeur K étant un multiple de N, on le remet à 1 et on ajoute 1 à I, la rangée ; puis l'on revient en boucle aux instructions précédentes pour loger le nouveau N1 (la valeur suivante de N1) dans la nouvelle cellule désignée.

Si K n'est pas un multiple de 1, on commence par lui ajouter 1 pour être à jour du compte, puis on établit I et J pour désigner la cellule en diagonale supérieure, et l'on se demande si on ne tombe pas en dehors du carré (ce qui se traduit par $I < 1$ ou $J > N$). Si non, on poursuit le processus en boucle ; si oui, on revient à l'opposé dans la colonne et la rangée, et le processus se poursuit.

Le programme respecte scrupuleusement cet organigramme. Les lignes 10 à 60 initialisent le système et la ligne 70 calcule le centre de la rangée. On attribue N1 à cette cellule (ligne 80) et on incrémente N1 de 1 (ligne 90). On se demande alors si N1 ne dépasse pas le nombre total de cellules (ligne 100) car alors, on aurait terminé le calcul du carré magique et sinon, si K est ou non un multiple de N (ligne 110). Si oui, on remet K à 1 (ligne 120) et on pointe la rangée suivante (ligne 130) pour repartir en boucle.

Si non (K n'est pas un multiple de N), on calcule la position de la cellule diagonale supérieure (lignes 160 et 170 et si I est différent de zéro (ligne 180), on passe à la ligne 210 car tout va bien ; mais si I = 0 (ligne 180, toujours), on transfère I à N, la rangée la plus éloignée. On corrigera J de la façon suivante (lignes 210 et 220) : si J est plus grand que N, on le ramène à 1.

Le processus étant achevé, on commande l'impression des résultats en veillant à bien répartir le carré, ce qui donne :

```
CARRE MAGIQUE DE 5 PAR 5

 17  24   1   8  15
 23   5   7  14  16
      6  13  20  22
 10  12  19  21   3
 11  18  25   2   9
```

La somme des cellules d'une rangée, d'une colonne ou d'une diagonale est bien 65 dans tous les cas !

Exercice :

Construisez, via l'ordinateur, un carré magique de 7 par 7.

Chapitre XIII

DU BASIC AU LSE

Il existe de très nombreux langages de programmation des ordinateurs, mais ils sont souvent adaptés à une gamme d'ordinateurs ou encore, à une application particulière.

Le micro-ordinateur LX 500 (515, 525,...) peut être programmé en Basic ou en LSE, si vous disposez de la disquette adéquate.

Attention !

Même si on utilise le même langage (Basic, par exemple), il y a toujours quelques différences entre les versions pour des ordinateurs différents. Mais la connaissance d'une de ces versions et la consultation de la notice suffisent à passer de l'une à l'autre.

Par exemple, il existe : *le BASIC N-8 ; le BASIC N-12 ; le BASIC S ; le BASIC G sur LX 500 ; les deux premiers sont standards mais proposent 8 ou 12 chiffres significatifs dans la mémorisation des nombres ; Basic S autorise des fonctions de calcul scientifique, Basic G dispose de facilités pour réaliser des programmes de Gestion.*

Autre exemple : *le LSE, langage encore jeune, s'améliore et plusieurs versions existent déjà : LSE sur T 1600, LSE sur MITRA 15, LSE sur LX 525, etc.*

La programmation, à l'aide d'un langage de programmation du type de LSE ou Basic, suppose que l'on sache :

1) *Avec quoi travailler ?* Il faut connaître la nature et la forme des données sur lesquelles on peut travailler (nombres ou chaînes de caractères) : représentation sur l'écran et le clavier, intervalle de définition, précision, ...

2) *Que peut-on faire ?* Quelles sont les opérations, les traitements que l'on peut effectuer sur ces données : opérateurs élémentaires, fonctions, procédures, ... ?

3) *Comment organiser ces traitements et en faire un programme ?* Il s'agit de la structure des instructions, des enchaînements et branchements, ...

Ce chapitre propose une réponse à ces trois questions, mais ne constitue qu'un début dans l'étude de la programmation.

★ ★
★

Le chapitre VIII nous a permis de construire petit à petit un programme de facturation, mais les instructions vous étaient imposées et, en fait, ce n'était qu'une introduction et quelques exemples d'utilisation d'instructions. Maintenant que vous êtes plus familiers avec l'ordinateur et ses commandes, nous allons formaliser un peu plus la « syntaxe » des instructions, la structure et l'écriture des données, et enfin, la structure générale d'un programme.

Dans les citations ou exemples, le langage LSE sera utilisé en premier *(ou à gauche de la page),* le Basic en second *(ou à droite de la page).* Mais vous ferez vite la différence : LSE utilise des mots français et Basic, des mots anglais.

1. STRUCTURE D'UN PROGRAMME

1.1. Un programme

Tout programme (LSE ou BASIC) se compose d'instructions numérotées (ou étiquetées) par un nombre compris entre 1 et ... (suivant les versions) 255, ou 32 000 et quelque. Même si les instructions ont été rentrées dans le désordre, *SEUL L'ORDRE NUMERIQUE DES ETIQUETTES COMPTE.*

La frappe de chaque instruction se termine par un code clavier : CNTRL plus S (en LSE) et RETURN (en Basic). La longueur d'une instruction Basic est limitée à une ligne d'écran (en général, 80 caractères), mais en LSE (et là encore, suivant les versions), elle peut atteindre 3 lignes d'écran.

On peut, à tout moment de la constitution d'un programme, le lister dans l'ordre des étiquettes en utilisant les commandes LISTER et LIST.

En LSE, on peut ne frapper que les deux premiers caractères des « commandes », l'ordinateur (ou plutôt l'interpréteur LSE) se charge de les compléter pour vous.

On obtiendra un listage du programme dans l'ordre et toutes corrections effectuées.

1.2. Une instruction

Elle se compose d'une étiquette en début de ligne (un numéro compris entre 1 et 255, ou 1 et 32 767 en LSE, entre 0 et 65 535 en Basic, mais ces dernières limites sont larges.

N'oublions pas qu'une instruction tient de la place en mémoire et que le LX ne dispose qu'au plus de 65 536 octets pour le système (LSE ou Basic), le programme, et les données.

Puis, après un blanc (en LSE), le corps de l'instruction se termine par un CNTRL + S ou RETURN.

Une *pseudo-instruction* doit être signalée tout de suite : elle sert à *documenter* le programme. C'est-à-dire à introduire des explications, des séparations, ...dans le corps du programme :

En LSE

```
n        *        texte.....
étiquette symbole  Ce texte sera reproduit à l'emplacement désigné par
          astérisque l'étiquette sans aucune influence sur l'exécution.
```

En Basic

```
n   REM   texte...
étiquette pour
          remarque
```

Les instructions doivent être écrites dans une forme très précise (une syntaxe) dont le détail est présenté au paragraphe 4. Elles comportent l'expression d'opérations et de traitements à opérer sur les données (constantes ou variables) et l'organisation de leur enchaînement.

2. REPRESENTATION DES DONNEES

« *Les ordinateurs calculent avec des nombres...* » C'est un fait bien commun. Mais :

• Avec quels nombres travaillent-ils ?

• Ne travaillent-ils pas aussi avec des « chaînes de caractères », de l'« alphanumérique » ?

Ces deux types de données sont fort différents, même si on peut les confondre facilement.

On doit donc faire une très grande distinction entre les nombres et les chaînes de caractères (même si elles ne comportent que des chiffres et des symboles de l'arithmétique) :

— Pour Z, R, =, $, ... pas d'ambiguité.

— Pour A, B, -, E, F, nous avons vu que certaines notations de codification (hexadécimal par exemple) les utilisent.

— Mais 3, 9, ..., sont-ils des caractères ou les chiffres d'un nombre ?

Seul le contexte, ou les « déclarations » le préciseront. *Mais c'est très important.*

2.1. Les constantes

Leur valeur est déclarée « en clair », elle figure dans le texte du programme :

```
F ← M * 1.176
CHAINE Z
Z ← ' BONJOUR '
```

```
L  = 2 * 3.14 * R
N$ = " DUPONT "
```

1.176, 2, 3.14 sont des *constantes numériques* (et la notation est identique en Basic et en LSE).

' BONJOUR ' (en LSE) et " DUPONT " (en Basic) sont des constantes chaînes de caractères.

2.1.1. Les constantes numériques

En LSE comme en Basic, les constantes numériques s'écrivent avec les conventions classiques à l'exception :

— de la virgule * qui est remplacée par un point ;

— du fait que les nombres, très grands ou très petits (en valeur absolue), sont affichés et imprimés en notation dite scientifique.

Dans ce cas, on n'affiche que quelques chiffres significatifs (6, 8, 9, 12 au plus, tout dépend du langage) suivis d'une puissance de 10 (écrite sous la forme E + 4, E − 3 par exemple). Ce facteur multiplicatif permet de « cadrer » le nombre de façon que la virgule vienne se placer derrière le premier chiffre significatif :

3,14	s'écrira	3.14	
314,18	"	3.1418	E 2
0,000314	"	3.14	E−4
3,1415926535	"	3.1415926	
		(en Basic 8)	

- *En LSE :* 9 chiffres significatifs sur le LX 500 (6 sur T 1600 et Mitra 15).
- *En Basic :* 8 chiffres en Basic 8 et 12 chiffres en Basic 12.

(*) Attention, certains langages, écrits en France, commencent à accepter les virgules !

En calcul de gestion, 12 chiffres sont nécessaires :

P = X XXX XXX XXX.XX

En calcul scientifique, 6 ou 8 suffisent souvent.

Mais les pertes de décimales dans les calculs successifs et les arrondis effectués sur la dernière décimale risquent fort de perturber la valeur affichée, aussi attention.

Exemple

Aussi bien en Basic qu'en LSE :

1 000 000 000 + 1 = 1 000 000 0000

Additionnez 1 autant de fois que vous voudrez !

En Basic 8 :

X + 1 = X
si X > 99999999.

De plus, même sous cette forme, l'amplitude autorisée pour les valeurs absolues numériques est bornée. Par exemple, en *Basic 8,* tout nombre inférieur à 10^{-63} est déclaré valoir 0 ; et tout calcul menant à une valeur supérieure à 9.99999999 10^{62} donne lieu à un « dépassement de capacité » et interruption du programme.

Mais 10^{62}... c'est déjà très grand.

En LSE, le dépassement de capacité survient vers $1{,}7.10^{38}$.

2.1.2. Les constantes « chaînes de caractères »

Ce sont des suites de caractères qui figurent explicitement dans le programme. Elles doivent être écrites entre apostrophes (LSE) ou guillements (Basic). Exemple :

```
AFFICHER 'AU REVOIR'          PRINT « AU REVOIR »
CHAINE Z                      A$ = A$ + " S "
Z = Z ! 'S'
```

Elles servent à afficher des textes, des libellés, des explications, dans les entrées-sorties au cours de l'exécution du programme. Elles n'ont pas besoin de déclaration préalable, les apostrophes ou les guillemets indiquant leur nature et leur longueur (leur longueur est manifestement limitée par la longueur de l'instruction).

2.2. Les variables

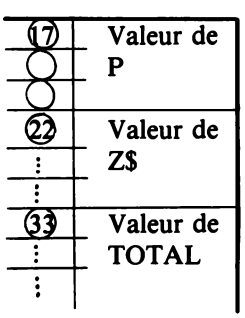
Ce sont des « entités » qui figurent dans le corps du programme et qui, à l'exécution, serviront à mémoriser, à un instant donné, une valeur alphabétique ou numérique.

Comme nous l'avons vu au chapitre VIII, la valeur contenue dans une variable peut changer au cours de l'exécution en fonction du déroulement du programme. En fait, l'identificateur d'une variable *(son nom)* représente l'adresse d'une zone de mémoire où la valeur de la variable sera enregistrée. La correspondance est établie par l'*interpréteur* à l'aide d'un tableau où figurent l'identificateur, le type de la variable et son adresse. Il construit ce tableau au fur et à mesure de l'introduction des instructions, mais le contenu de ces zones mémoires sera modifié au cours de l'exécution du programme.

Par exemple :

P	Nbre	17
Z$	Chaîne	22
TOTAL	Nbre	33

table des identificateurs (à disposition de l'interprèteur)



Mémoire

Les identificateurs s'écrivent avec :

LSE	*Basic*
Au plus 5 caractères, le premier étant une lettre !	Une seule lettre, éventuellement suivie d'un chiffre.
Les chaînes de caractères doivent être « déclarées »	Les identificateurs de chaînes sont suivis du symbole $.
Exemples TOTAL, SOM	T , S
CHAINE NOM	N$

Une remarque s'impose sur l'occupation mémoire que nécessite ces variables. En effet, les *constantes* figurent dans le texte du programme et donc, n'occupent que cette place-là. Les *variables,* par contre, ne figurent dans le programme que par leur identificateur et leur valeur est mémorisée ailleurs, d'où une occupation de place supplémentaire. Un nombre occupe de 5 à 7 octets de la mémoire centrale (5 en Basic S et LSE ; 7 en Basic 12). Une chaîne occupera 1 octet par caractère (mais parfois 1 octet de plus, suivant les interpréteurs).

La longueur d'une chaîne est déterminée :

• En Basic, soit par une déclaration du programmeur (voir l'instruction DIM), soit, par défaut, à la valeur « 10 » lors de la première rencontre de l'identificateur (et on ne doit plus dimensionner cette variable).

• En LSE, cette longueur est « dynamique » et une chaîne de caractères n'occupera en mémoire que sa longueur réelle à l'instant présent.

Remarque :

En Basic, seule est traitée la longueur vraie d'une chaîne, mais le volume mémoire réservé sera toujours égal à 10 (par défaut) ou à la valeur demandée. Donc attention lors des déclarations ! Le volume mémoire est limité.

2.3. Les tableaux

Très souvent, on a besoin de faire subir le même traitement à plusieurs valeurs distinctes d'une même notion : les prix unitaires de plusieurs produits, les

âges de chacun des élèves d'une classe, ... Plutôt que de donner à ces valeurs des identificateurs différents A, B, C, ... (ce qui nous obligerait à écrire autant de séquences d'instructions qu'il y a de valeurs), on donne à cet ensemble de données *un seul identificateur* mais on précise la donnée par un indice placé entre parenthèses en Basic, et entre crochets en LSE. Par exemple :

- En LSE : AGE [4] *(4e valeur du tableau des âges).*

- En Basic : P (8) *(valeur du prix unitaire du produit d'indice 8).*

Mais cet indice lui-même peut être une variable et nous permet d'écrire une seule instruction pour traiter un quelconque élément du tableau : il nous suffira de faire varier l'indice :

```
... AGE [I] ...
...   P (I)  ...
```

On pourra même remplacer l'identificateur d'indice par une expression arithmétique simple. Par exemple :

P (2 * I + 1)

permet d'obtenir toutes les valeurs de rang impair quand I vaut 0, puis 1 puis 2,...

Toutefois, attention :

- en LSE, l'indice varie de 1 à N ;
- en Basic, l'indice varie de 0 à N ;

(N étant la dimension du tableau).

On peut aussi utiliser des tableaux à 2 indices, l'habitude étant que le 1er indice désigne la ligne et le 2e, la colonne dans une représentation du tableau sur 2 dimensions. Par exemple :

4	5	8	6	4
2	1	0	−4	2
−6	0	3	5	1

Pour cette matrice nommée T :

- en LSE T [I , J] $1 \leqq I \leqq 3$; $1 \leqq J \leqq 5$

• en Basic T (I ; J) $0 \leqq I \leqq 2$
$0 \leqq J \leqq 4$

Remarque :

Lorsqu'il n'y a pas de problème de place mémoire, on peut toujours, même en Basic, n'utiliser que les valeurs de l'indice I non nulles, afin que la valeur de l'indice corresponde effectivement au numéro de ligne et/ou de colonne : la 1re ligne et la 1re colonne du tableau en mémoire sont alors inutilisées.

• En Basic, les éléments d'un tableau sont obligatoirement des *NOMBRES.*

• Les plus récentes versions de LSE acceptent les tableaux de *CHAÎNES* (consultez la notice).

3. LES OPERATEURS DE TRAITEMENT

Les données (variables ou constantes) étant définies et leur valeur supposée présente en mémoire, que peut-on en faire, comment peut-on les traiter en vue d'obtenir des résultats intermédiaires ou des résultats terminaux à « sortir » sur l'écran ou l'imprimante ? Commençons par les opérateurs numériques (les plus classiques).

3.1. Opérateurs numériques

Ils vont travailler sur les nombres (variables et/ou constantes numériques) et le résultat sera une variable numérique.

3.1.1. Symboles des opérateurs élémentaires

En LSE, comme en Basic, on note :

« + » pour l'addition ;
« * » pour la multiplication ;
« / » pour la division ;
« − » pour la soustraction ;

mais aussi « − » pour le changement de signe ; exemple : − 3.

LSE propose, en plus, le symbole « ↑ » pour l'élévation à la puissance : Z = X ↑ 2 (X au carré).

Attention à A ↑ X :

• Si X est entier, il y a X multiplications successives.

• Si X n'est pas entier, on passe par logarithme et exponentielle, donc A doit être positif !

Ces symboles élémentaires peuvent être composés dans des expressions algébriques classiques utilisant les parenthèses, avec toutes les lois de l'algèbre classique, mais tout s'écrit sur une seule ligne :

$$Z \leftarrow ((X + 2) * (5 + Y \uparrow 2)) / (A + B)$$

pour :

$$z = \frac{(x+2) \cdot (5+y^2)}{(a + b)}$$

3.1.2. Les fonctions numériques

Certaines fonctions plus complexes s'expriment non plus comme des opérateurs placés entre deux variables mais à l'aide de *MOTS-CLES,* la variable étant placée entre parenthèses. Nous en avons l'habitude avec sin (x). Voici les principales :

LSE	BASIC	Signification
RAC (...)	SQR (...)	Racine carrée
ABS (...)	ABS (...)	Valeur absolue
ENT (...)	INT (...)	Partie entière
ALE (...)	RND (...)	Génération d'un nombre pseudo-aléatoire

Mais aussi :

LSE	Basic S	Signification
SIN	SIN	Sinus
COS	COS	Cosinus
ATG		Arc tangente
LGN	LOG	Logarithme népérien
EXP	EXP	Exponentielle

3.2. Opérateurs sur chaînes

3.2.1. La concaténation

On utilise un seul symbole d'opérateur sur les chaînes : la « concaténation », mot fort savant qui désigne la réunion de deux chaînes adjacentes en une seule. Par exemple :

AFFICHER ' BONJOUR ' ! ' MONSIEUR '

```
X$ = " BONJOUR "
Y$ = X$ + " MONSIEUR "
```

LSE utilise le « ! » (point d'exclamation du clavier), Basic utilise le « + » (plus numérique) ; cet opérateur travaille entre deux chaînes (constantes et/ou variables) et le résultat est une nouvelle chaîne. Les autres opérateurs sur chaînes s'expriment à l'aide de fonctions sur chaînes ou de notations particulières.

3.2.2. Les sous-chaînes

On appelle « sous-chaîne » une partie d'une chaîne donnée (éventuellement réduite à 1 caractère). En Basic, on utilise une notation indicée sur l'identificateur de la chaîne : A$ est une chaîne ; A$ (I ; J) est la sous-chaîne comportant les caractères dont les rangs vont de I à J compris ; A$ (I) est la sous-chaîne du rang I à la fin réelle de la chaîne. *Par exemple* A$ (X ; X) *isole le* X^e *caractère de la chaîne.*

En LSE, on utilise la fonction sous-chaîne SCH. Elle s'écrit SCH (C, I, X) avec C : chaîne origine ; I : rang du 1er caractère ; X : si X est numérique, c'est la longueur de la sous-chaîne ; si X est alphabétique, sa valeur est un ensemble de caractères d'arrêt.

3.2.3. Autres fonctions sur chaînes de caractères

LSE	*BASIC*	Signification
LGR (...)	LEN (...)	Longueur d'une chaîne
CNB (...)	VAL (...)	Conversion d'une chaîne en nombre
CCA (...)	STR$ (...)	Conversion d'un nombre en chaîne
EQN (...)	ASC (...)	Equivalent numérique d'un caractère
EQC (...)	CHR$ (...)	Equivalent en caractère d'une valeur numérique comprise entre O et 255

Par exemple :

AFFICHER EQC *(n)*

ou PRINT CHR$ *(n)*

permet d'envoyer à la console des codes spéciaux pour manipuler sur l'écran :

n = 7 déclenche une sonnerie
n = 26 efface l'écran

LSE offre bien d'autres possibilités d'opérateurs sur chaînes et d'options dans les précédentes mais leur étude sort d'une première présentation. Citons cependant les possibilités de recherche de la présence et de la position de sous-chaînes et/ou de caractères.

3.3. Les expressions

Tout comme en algèbre, les deux langages acceptent l'écriture d'expressions comportant constantes, variables et fonctions reliées par des opérateurs, utilisant les parenthèses dans la convention classique.

Les expressions s'écrivent sur une seule ligne, aussi doit-on utiliser les parenthèses pour remplacer l'écriture classique sur plusieurs niveaux :

$$z = \frac{\sin(x)}{\cos(x)}$$ s'écrira Z ← SIN (X) (COS (X)

$$z = \frac{1}{\sin(x) + \cos(x)}$$ Z ← 1 / (SIN (X) + COS (X))

Elles peuvent être aussi bien numériques, qu'alphanumériques :

```
CHAINE A, B
B ← '.....'                    B$ = ".....".
A ← A ! SCH (B,1,20-LGR(A) )   A$ = A$ + B$ (1 ; 20-LEN(A$) )
```

Cette expression complète la chaîne A (A$) à 20 caractères avec des « . »

L'exécution d'une expression respecte les lois de priorité suivantes :

1 — exécution des parenthèses les plus intérieures ;
2 — exécution des fonctions ;
3 — exécution du − (changement de signe) et de l'exponentielle (↑ en LSE) ;
4 — exécution de * et / ;
5 — exécution de + et − .

A chaque niveau, les priorités égales s'exécutent de gauche à droite.

4. STRUCTURE DES INSTRUCTIONS

4.1. Les déclarations

Si, en Basic, la nature des variables est portée par l'identificateur lui-même (Z est un nombre, A$ une chaîne alphanumérique), il n'en est pas de même en LSE et, de toutes façons, la taille des tableaux doit être « déclarée ». On trouvera donc quelques instructions de déclaration : *elles doivent précéder tout emploi de la variable et ne figurer qu'une fois dans le programme.*

LSE	*BASIC*
Déclaration de chaîne :	
n CHAINE NOM, ADRS	inutile
Déclaration de longueur maximale d'une chaîne	
inutile	n DIM A$ (30)
Déclaration de tableau	
n TABLEAU Z [25]	n DIM Z (25)
n TABLEAU X [3,5]	n DIM X (3 ; 5)
Attention : cette déclaration ouvre le tableau Z à 25 positions ($1 \leq I \leq 25$) et le tableau X à 3 * 5 = 15 positions.	Attention : cette déclaration ouvre le tableau Z à 26 positions ($0 \leq I \leq 25$) et le tableau X à 4 * 6 = 24 positions.

4.2. L'affectation

La mémorisation de résultats (intermédiaires ou définitifs) s'effectue grâce à des identificateurs auxquels on « affecte » la valeur calculée :

CHAINE X , TRUC	
TOTAL ← MONTANT + PORT	T = M + P
X ← TRUC ! 'S'	X$ = T$ + " S "

(La flèche dirigée à gauche ne figure pas sur tous les claviers, elle est alors remplacée par le « souligné ».)

L'expression qui se trouve à droite est effectuée et la valeur résultante attribuée à l'identificateur de gauche (toute ancienne valeur de cet identificateur est donc détruite).

4.3. Les instructions d'entrée-sortie

L'affichage de résultats sur l'écran et la saisie de données au clavier, s'exécutent avec les instructions AFFICHER/PRINT, et LIRE/INPUT.

4.3.1. Affichage de résultats... ou de questions !

N'oubliez pas qu'un des gros avantages du micro-ordinateur est de fonctionner en « conversationnel » et que donc, au cours de l'exécution d'un programme, il peut y avoir, par exemple :

- affichage d'une question à l'écran ;
- saisie de la réponse au clavier ;
- traitement de cette réponse ;
- affichage d'un résultat ;
- etc.

L'instruction AFFICHER/PRINT suivie d'une liste de variables, de constantes ou d'expressions, séparées par des virgules (LSE) ou des point-virgules (Basic) :

— effectue les expressions et affiche la valeur ;
— affiche la valeur des variables ;
— affiche la constante ;

dans l'ordre de la liste, avec un blanc (et le signe « – » éventuellement) devant les nombres.

Quelques fonctions de type FORMAT * (en LSE) ou TAB ** (en Basic) permettent de faire de la belle mise en page, mais... c'est plus compliqué.

4.3.2. Entrée de données

Disons tout d'abord que l'exécution des instructions d'entrée LIRE ou INPUT déclenche : un « tûûûût » sonore en LSE et l'affichage d'un « ? » en Basic, après quoi le clavier est débloqué (car en temps d'exécution normale, il est bloqué, ou du moins, aucune saisie, aucun affichage n'a lieu. Essayez en Basic 10 GOTO 10, et frappez au clavier. Alors, l'ensemble des caractères frappés au clavier sera affecté à l'identificateur qui suit l'instruction LIRE, ou l'INPUT. L'introduction de la donnée se terminera par un :

— RETURN systématique en Basic ;
— CNTRL + S en LSE pour une chaîne de caractères ;
— un « blanc » ou un « return » ou encore un CNTRL + S pour un nombre en LSE.

Basic permet de remplacer le « ? » par un petit libellé, tout simplement en faisant suivre l'instruction INPUT par une constante chaîne. Par exemple :

```
INPUT " QUEL EST VOTRE NOM ? : " ; N$
```

En LSE, on fera précéder le LIRE d'un AFFICHER, suivi d'un point-virgule.

4.4. Les branchements

L'ordre d'exécution normale d'un programme est celui de l'ordre des numéros d'instructions (ou étiquettes). Mais deux instructions peuvent rompre cette « séquence *** ».

(*) Formater la ligne.

(**) Pour tabulation : TAB (23) = « en colonne 23 ».

(***) Voir le *Petit Robert :* « Suite ordonnée ».

4.4.1. Le branchement inconditionnel

Il donne l'étiquette de l'instruction suivante :

n ALLER EN *m* *n* GOTO *m*

et peut servir à se « brancher » à une séquence d'instructions implantée plus loin, ou avant, dans le programme. Cette instruction est exécutée *inconditionnellement* dès qu'on la rencontre.

4.4.2. Le branchement conditionnel

Il est beaucoup plus important car il exécute des branchements suivant le résultat d'un test. Son écriture elle-même en donne l'explication :

• En LSE :
SI *(une condition)* ALORS *(une instruction)* SINON *(une autre instruction).*

• En Basic :
IF *(une condition)* THEN *(une instruction)* ELSE *(une autre instruction).*

Qu'est-ce qu'une condition ? C'est une expression qui utilise un symbole logique entre deux variables et/ou constantes ; donc, le résultat est OUI ou NON. Les symboles autorisés sont au nombre de 6 :

$$= \ , \ > \ , \ < \ , \ = > \ , \ < = \ , \ < > \ ^{*}$$

et leur signification évidente :

A = 3 X$ = " OUI "
Z > 19 Y > = 4

> ***Remarque***
>
> *On pourra même utiliser certains opérateurs logiques (ET, OU en LSE), (AND, OR en Basic) entre deux conditions pour les composer*
>
> *A* > 8 AND A < = 12

(*) Consultez la notice de votre machine !

Les instructions qui suivent les mots ALORS, THEN, SINON et ELSE pourront être, suivant le langage :

— une instruction simple ;
— un ALLER EN ;
— une instruction composée, c'est-à-dire une suite d'instructions séparées par des point-virgules commençant par le mot DEBUT et se terminant par le mot FIN.

— une instruction simple ;
— une instruction GOTO :
— ou même une simple étiquette.

On peut alors réaliser toutes les organisations que l'on souhaite, d'autant que les SINON et ELSE peuvent être omis, auquel cas, si la réponse au test est non, on passe à l'instruction suivante.

4.4.3. L'arrêt, la pause

L'éxécution normale d'un programme peut être interrompue par plusieurs faits :

• Les instructions TERMINER et END stoppent l'exécution et mettent l'interpréteur en attente d'une commande.

• Les instructions PAUSE et STOP ont le même effet, *mais* on peut reprendre le cours de l'exécution par deux commandes : Continuer et RESUME, à l'endroit précis de l'arrêt et sans perdre les valeurs des variables.

• Une erreur de syntaxe : erreur fatale, car il faudra redemander l'exécution après correction de l'erreur.

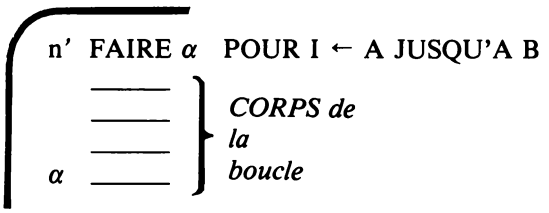
• Une commande d'arrêt : CNTRL + X ou ESCAPE en LSE, CNTRL + C en Basic.

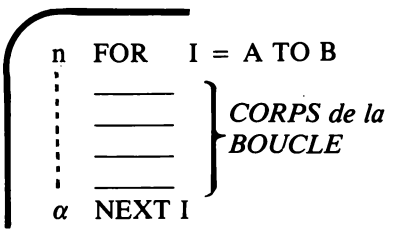
A l'issue d'un arrêt par TERMINER, PAUSE et STOP, l'interpréteur donne le numéro de ligne où s'est produit l'arrêt.

Quelqu'autres instructions et commandes permettent de reprendre à un endroit donné ou de créer des arrêts artificiels, mais...

4.5. La boucle

Bien que cette instruction ne soit pas indispensable et qu'elle soit même souvent mal utilisée, il est bon de la donner car (surtout en Basic) elle permet l'accélération de l'exécution de parties répétitives dont on connaît le nombre d'exécutions demandées. La forme la plus simple est :





L'indice I peut être utilisé dans le corps de la boucle comme variable ou indice de tableau.

Le corps de la boucle, c'est-à-dire l'ensemble des instructions qui suivent FAIRE (ou FOR), et jusqu'à l'instruction d'adresse α comprise (ou NEXT I), sera exécuté pour I = A, puis I = A + 1, puis ... jusqu'à ce que I soit supérieur à B ; après quoi, l'instruction suivante sera celle qui suit α (ou NEXT I).

Là encore, suivant les langages et les versions, il est des formes de boucles plus complexes, mais dont l'emploi n'est pas toujours optimal.

5. LES PROCEDURES ET SOUS-PROGRAMMES

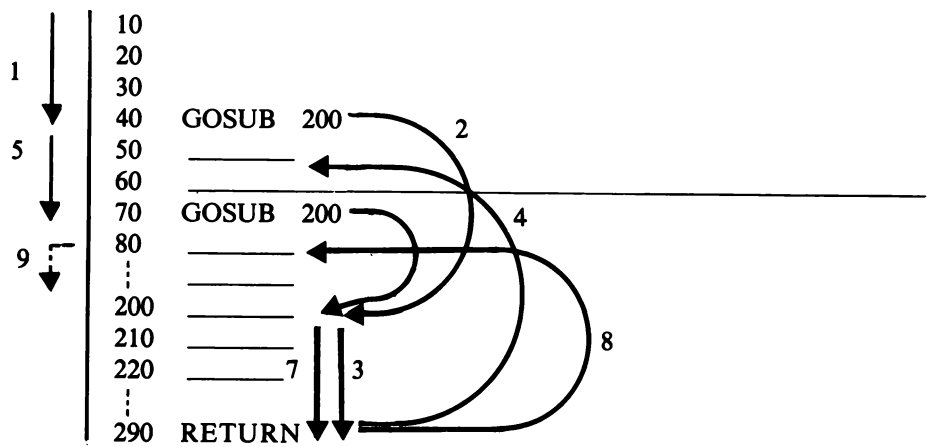
Parfois, certaines séquences d'instructions, certains morceaux de programme doivent être utilisés plusieurs fois mais sur des données différentes et à divers emplacements du programme. On peut, bien sûr, écrire la suite des instructions autant de fois qu'il est nécessaire (et il est même recommandé d'avoir des petits morceaux de programme tout prêts que l'on recopiera en cas de besoin).

La notion de sous-programme et celle, plus générale de procédure, va nous éviter cette peine. Si un morceau de programme doit être utilisé plusieurs fois, on ne l'écrira qu'une fois dans le programme, on l'isolera en quelque sorte et on y fera appel en cas de besoin.

5.1. En Basic

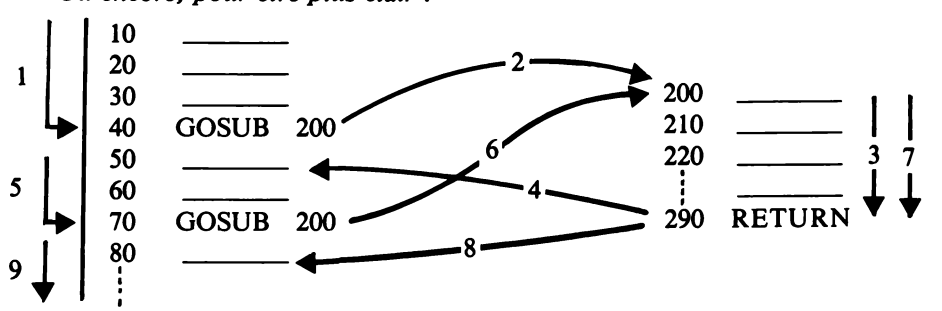
La séquence d'instructions sera écrite quelque part dans le programme à une adresse « *ad* » quelconque et se terminera par l'instruction RETURN *(Retourne, reviens à l'endroit d'où on a appelé* le SOUS-PROGRAMME).

Dans le programme lui-même, on fera « appel » à ce sous-programme par l'instruction : GOSUB *ad.* C'est un GOTO, mais un GOTO qui demande au système de mettre de côté l'adresse d'où l'on « part », en quelque sorte, pour pouvoir y revenir, et ce à la rencontre du RETURN. Un petit graphique vaut mieux que de longs discours :



• *La séquence 200 à 290 sera donc exécutée 2 fois, mais le* « retour » *se fait toujours à l'instruction qui suit le* GOSUB *d'appel.*

Ou encore, pour être plus clair :



5.2. En LSE

En LSE, on utilise la notion de PROCEDURE : on va donner un nom à cette séquence d'instructions et, pour bien le distinguer d'un identificateur, on la précède du symbole « & ».

En Basic, les identificateurs doivent porter le même nom dans le programme et le sous-programme, mais en LSE on peut employer des identificateurs différents ; le seul problème sera donc de faire correspondre les valeurs des données et des résultats.

Deux cas :

• Plusieurs valeurs sont à récupérer, et on procède par une liste d'identificateurs à écrire dans la définition de la procédure et dans son appel.

• La procédure ne calcule qu'une seule valeur en résultat et on utilisera la procédure comme une fonction.

Voyons deux exemples.

Procédure

Programme principal

```
_______
_______ lire T1, T2, T3
& TOTO (T1, T2, T3, TMOY, TMAX)
_______
_______ utilisation de TMOY et de TMAX.
```

Procédure

```
PROCEDURE  & TOTO (X1, X2, X3, MOY, MAX)
_______
_______ Calcul de la moyenne MOY et de la
_______ valeur maximale de X1, X2, X3
MOY ← ...
MAX ← ...
RETOUR  (renvoi à l'instruction qui suit l'appel et
         remplace TMOY et TMAX par les valeurs
         de MOY et MAX)
```

Procédure - Fonction

Programme principal

```
_______lire T1, T2, T3
_______
TMOY ← & MOY (T1, T2, T3)
_______
_______
```

Procédure fonction

```
PROCEDURE & MOY (X1, X2, X3)
Z ← (X1 + X2 + X3) / 3
RESULTAT Z   (remplace & MOY (...) par Z et renvoie
              à l'instruction d'appel).
```

Le gros avantage de ces procédures est de pouvoir écrire des « sous-programmes » tout à fait indépendants du programme principal et du choix de ses identificateurs.

6. INITIALISATION DE L'ORDINATEUR ET MANIPULATION DU TERMINAL

L'emploi d'une disquette LSE ou d'une disquette Basic diffère au niveau des commandes et ce chapitre présente, en parallèle, les commandes les plus utiles dans les deux langages.

A l'initialisation (LE bouton de façade) :

LSE demande la :	*Basic affiche :*
DATE XX^XX^XX O	BDOS ...
	?
On doit alors déclarer :	*Il faut alors appeler le* Basic *par*
BO x	? B ; BASIC X
pour BONJOUR *	*et obtenir*
	xx BASIC N ...
PRET	COMMAND :

On se trouve alors :

sous LSE	*sous* Basic.

(*) Ceci, afin de rester compatible avec d'autres versions LSE !

LSE	Basic	Commentaires
APPELER xyz	LOAD xyz	Charger en mémoire le programme xyz.
LISTER A PARTIR de n, n_2	LIST n ; n^2	Lister le programme à partir de n_1 et jusqu'à n_2.
EXECUTER A PARTIR de n	RUN n	Exécution à partir de l'instruction n (1 par défaut).
NUMERO A PARTIR de n_1, n_2	NUMB n_1 ; n_2	Renuméroter les lignes en partant de n_1 et de n_2 en n_2.

etc., car il existe de très nombreuses commandes qu'il n'est pas question de décrire ici.

Enfin, on terminera l'utilisation du terminal sous LSE *par :*
AU... REVOIR

★ ★
★

En conclusion de ce chapitre

Nous vous avons présenté les principaux éléments de deux langages : le LSE et le Basic ; à l'aide de ces instructions, vous pouvez déjà écrire quelques programmes, mais n'oubliez jamais que la programmation nécessite, pour devenir efficace, une étude approfondie de la structure des problèmes (nous l'appellerons l'*analyse*) et du langage particulier de votre ordinateur.

Chapitre XIV

QUELQUES PROGRAMMES PRATIQUES EN LSE

De la même façon que nous avons donné quelques programmes-types en Basic, nous en présentons ici en LSE, « langage symbolique d'enseignement ». Il s'agit de montrer l'emploi pratique du LSE à partir de problèmes simples, et jusqu'à un jeu : le jeu de Marienbad (*).

1. BON COURAGE

Il en faudra peu, au lecteur, pour suivre ce court programme d'introduction... Mais les programmes suivants seront plus consistants !

Langage d'enseignement, le LSE a le mérite aussi d'être en français, ce qui en rend son abord plus aisé (pour des francophones...). Le court programme suivant va le montrer.

Il commence par une « déclaration » de chaîne, c'est-à-dire par une instruction qui n'est pas exécutable mais dont l'unique objet est de renseigner le système qui réservera alors en mémoire la place nécessaire. Cette déclaration, c'est celle de « CHAINE PR », où PR (pour « prénom »), caractérise la « chaîne de caractères » formant le prénom retenu. En supposant que cette ligne vienne en n° 1, l'ensemble du programme est :

(*) Les programmes proposés ici ont été repris du « Manuel d'utilisation LSE », numéro spécial du bulletin de liaison de l'*Informatique dans l'enseignement secondaire,* INRP, janvier 1975, publié sous l'autorité de M.C. Lafond, avec l'aimable autorisation de l'Institut National de Recherche Pédagogique.

```
1 CHAINE PR
3 AFFICHER 'TAPEZ VOTRE PRENOM : ';LIRE PR
5 AFFICHER 'BON COURAGE, ',PR,', POUR L''ETUDE DU L S E !'
7 TERMINER
```

La ligne 3 commande deux choses : tout d'abord, l'affichage du texte placé entre deux apostrophes et qui est, ici, TAPEZ VOTRE PRENOM ; puis, la lecture du prénom dont la machine a demandé la frappe à l'opérateur.

La ligne suivante (ligne 5) va utiliser ce prénom en l'introduisant dans une phrase complète selon le même principe, la ligne 7 indiquant la fin du programme.

Le lancement de ce programme se fait à l'aide de la commande :

```
EXECUTER A PARTIR DE 1
```

qui sera fournie à la machine en tapant uniquement EX suivi du caractère de fin de ligne ; l'ordinateur complète la phrase et affiche EXECUTER A PARTIR DE, après quoi, il suffira de frapper le numéro de la ligne.

On lira alors, en supposant que vous frappiez votre prénom au clavier :

```
TAPEZ VOTRE PRENOM : CORALIE
BON COURAGE, CORALIE, POUR L'ETUDE DU L S E !
TERMINE EN LIGNE 007
```

La formule de la dernière ligne dépend, en pratique, du type de machine sur laquelle est exécuté le programme et pourra par conséquent être différente.

2. CALCUL DE L'AIRE DE TRIANGLES

Un peu plus difficile, cette fois, ce programme calcule la surface de triangles et introduit des instructions de branchement inconditionnel et de listage.

Soit à calculer l'aire d'un triangle, connaissant la base B et la hauteur H. Tout le monde connaît la formule à appliquer : la surface est égale à S = (BH) / 2.

Le programme à rédiger reste très simple. Il commence par quelques instructions « AFFICHER » qui exposent son mode d'emploi complet, ce qui donne :

```
1 AFFICHER 'CE PRØGRAMME PERMET DE CALCULER L"AIRE DE PLUSIEURS TRIANGLES'
2 AFFICHER 'VØUS DØNNEREZ LA VALEUR DE LA BASE,DEMANDEE CØMME SUIT:'
3 AFFICHER 'B='
4 AFFICHER 'ET CELLE DE LA HAUTEUR, DEMANDEE CØMME SUIT :
5 AFFICHER 'H='
6 AFFICHER 'SI B=67 ET H=43 PAR EXEMPLE,LA SØLUTIØN SERA DØNNEE SØUS LA FØRME'
7 B←67;H←43;ALLER EN 9
8 AFFICHER 'B=';LIRE B;AFFICHER 'H=';LIRE H
9 S←B*H/2
10 AFFICHER 'S=',S
11 AFFICHER ''
12 ALLER EN 8
```

Après calcul et affichage, la dernière ligne (12) ne commande pas la fin du programme mais renvoie à la ligne 8 : vous pourrez ainsi exécuter un nouveau calcul.

Le lancement du programme se fait comme précédemment et donne les résultats suivants, où l'on a souligné ce que l'opérateur devait réellement frapper :

```
EXECUTER A PARTIR DE 1

CE PRØGRAMME PERMET DE CALCULER L"AIRE DE PLUSIEURS TRIANGLES
VØUS DØNNEREZ LA VALEUR DE LA BASE,DEMANDEE CØMME SUIT:
B=
ET CELLE DE LA HAUTEUR, DEMANDEE CØMME SUIT :
H=
SI B=67 ET H=43 PAR EXEMPLE,LA SØLUTIØN SERA DØNNEE SØUS LA FØRME:
S=1440.5

B=34
H=12
S=204

B=86.4
H=23.009
S=993.9886

B= (ESC)
PRET EN LIGNE 008
```

On voit qu'on a exécuté deux exercices consécutifs, l'un avec B = 34 et H = 12 et l'autre avec B = 86,4 et H = 23,009. Après quoi, l'ordinateur obstiné demandant des données pour un troisième, on a actionné la touche d'interruption ESC. L'ordinateur est alors « sorti » du programme non sans afficher le numéro de la ligne où il en était resté (ligne numéro 8).

Exercices :

1. Rédigez un programme calculant la surface d'un cercle connaissant son rayon.

2. Rédigez un programme calculant la surface d'une sphère. (Tiens, tiens, ... hésiteriez-vous ?)

3. RANGEMENT DE NOMBRES PAR ORDRE CROISSANT

On retrouve ici un problème parfaitement classique, qui a été développé précédemment en Basic. Voici l'une de ses solutions en LSE.

On dispose d'une liste de nombres, dans le désordre, et on souhaite que l'ordinateur les présente dans l'ordre croissant.

Pour cela, on va tracer un ordinogramme traduisant l'algorithme adopté qui procède de façon tout à fait classique par comparaisons et permutations. Cet ordinogramme est donné dans la figure 14-1.

La liste (le « tableau ») comprend N nombres, repérés par la lettre A ; ainsi, on aura A(1) pour le premier, A(2) pour le second, etc., jusqu'à A(N). La méthode consiste à comparer A(1) à A(2) : si A(2) est plus grand que A(1), ils ne changent pas de place ; mais si A(2) est plus petit, on les permute. Désormais, le nouvel A(1) est plus petit que le nouvel A(2).

Afin de permuter (éventuellement) A(1) et A(2), on va introduire une variable auxiliaire notée AUX qui intervient ainsi :

1) On transfère A(1) dans AUX ;
2) On fait passer l'ex-A(2) dans A(1) ;
3) Et le contenu de AUX est transmis au nouvel A(2).

Puis, on compare le nouvel A(2) à A(3) en procédant rigoureusement de la même façon et ce, jusqu'au bout du tableau.

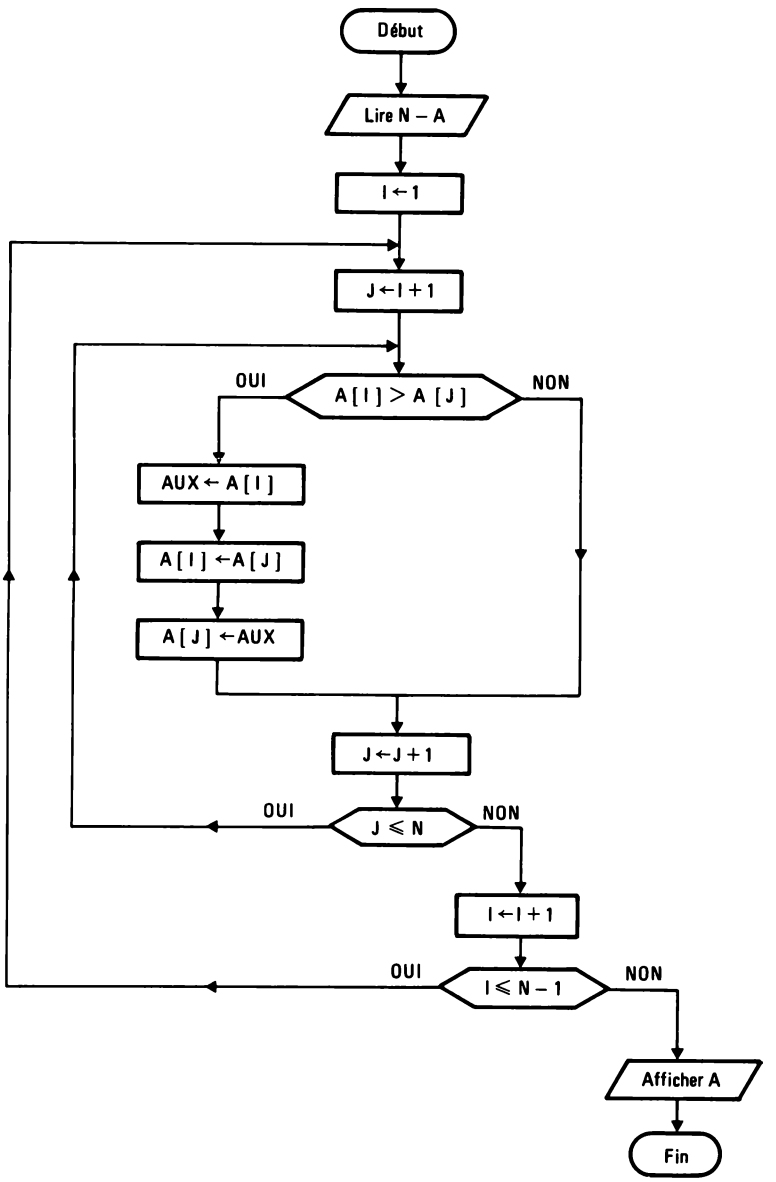


Fig. 14.1 — Organigramme de rangement de nombres par ordre croissant.

Cette première série de comparaisons et de permutations éventuelles n'aura cependant que commencé à remettre de l'ordre dans la liste, mais il faudra d'autres « passes » de ce type pour la remettre complètement dans l'ordre. En effet, supposons que N = 4 et que les nombres soient 3, 6, 5 et 4 ; le processus est le suivant, pour une première passe :

- On compare 3 à 6 : ils sont dans l'ordre.
- On compare 6 à 5 : le nombre 5 étant plus petit, on le permute avec 6.
- La nouvelle liste est donc 3, 5, 6, 4.
- On compare ensuite 6 avec le dernier chiffre, 4. Ce dernier étant plus petit, on permute à nouveau.
- A la fin de la première « passe », la liste est : 3, 5, 4, 6.

On constate bien que le classement n'est pas terminé ! Aussi faudra-t-il exécuter une seconde, et peut-être davantage de passes, leur nombre étant de N – 1 avec N, le nombre d'« objets » à classer.

On peut aussi constater que la plus petite valeur a été chassée, dès la première passe, au début de la liste d'où elle ne bougera plus. En toute rigueur, on pourra donc se permettre de réduire de un le nombre de comparaison à chaque passe successive.

Si l'on suppose que l'opérateur n'introduira les valeurs à classer qu'après lancement du programme et sur sa demande, le programme correspondant est le suivant :

```
1*RANGEMENT DE N NOMBRES DANS UN TABLEAU DANS L'ORDRE CROISSANT
3 AFFICHER 'N= ';LIRE N;TABLEAU A(N)
5 AFFICHER 'LECTURE DES N NOMBRES : ';LIRE A;AFFICHER ''
7 FAIRE 13 POUR I=1 JUSQUA N-1
9 FAIRE 13 POUR J=I+1 JUSQUA N
11 SI A(I)>A(J) ALORS DEBUT AUX=A(I);A(I)=A(J);A(J)=AUX FIN
13
15 AFFICHER A;TERMINER
```

A la ligne 1, l'astérisque indique qu'il s'agit d'un commentaire et non d'une instruction. La ligne 3 demande à l'opérateur la frappe du nombre N d'objets à classer et définit ensuite l'importance du tableau, renseignement indispensable à la machine.

La ligne 5 commande l'affichage des nombres à classer, frappés par l'opérateur. Puis, ces nombres A(1), A(2), etc., étant généralisés sous l'appellation A(I), avec I pouvant prendre toutes les valeurs de 1 à N, on va exécuter une première passe, laquelle comprend obligatoirement N – 1 comparaisons. De ce fait, on limitera I à N – 1.

La seconde boucle (ligne 9) prépare les comparaisons entre les valeurs I + 1 à N, cette fois, en appelant systématiquement J la somme I + 1.

En se référant au programme, on voit que ces deux boucles (lignes 7 et 9) renvoient à la ligne 13, vide. Son rôle consiste uniquement à définir la *fin de boucle ;* le système sait ainsi que la fin de boucle est atteinte , qu'il faut ajouter le pas défini au préalable (1, ici), à la variable contrôlée et faire le test qui suit.

La ligne 11 est typique du LSE : pour qu'une permutation soit exécutée, il faut que A(I) soit plus grand que A(J), donc que A(I + 1). C'est la condition posée par « SI » ; mais dans ce cas, tout ce qu'il convient de faire est indiqué sur la même ligne, entre les mots-clés DEBUT et FIN. C'est ce qu'en LSE, on appelle une *instruction composée.*

Lançons le programme ; on obtient (en tenant compte du fait que ce qui est souligné ici a dû être introduit au clavier par l'opérateur) :

```
EXECUTER A PARTIR DE 1

N= 10
LECTURE DES N NOMBRES : 21.7 13.5 -54.2 -28 13.5 35 57.6 -12 7.25 18

-54.2
-28
-12
7.25
13.5
13.5
18
21.7
35
57.6

TERMINE EN LIGNE 015
```

Cette méthode, qui procède en « faisant monter » la plus faible valeur au début de la liste, est appelée « méthode de la bulle ».

Exercice :

Introduisez dans ce programme un test vérifiant que le nombre de valeurs proposées est bien égal au total N précédemment frappé.

4. CE NOMBRE EST-IL PREMIER ?

Trouver si un nombre est premier est aisé lorsque ce nombre est petit. Mais s'il est grand ? Grâce à sa vitesse de travail, l'ordinateur résoudra pour vous ce problème en un clin d'œil.

Dans ce programme, on va introduire des fonctions qui n'ont pas encore été rencontrées jusqu'à présent. Elles n'épuisent pas, pour autant, toutes celles qu'offre le LSE...

Le problème est le suivant : soit un nombre entier N ; est-il premier ? Si oui, la machine va l'indiquer ; si non, elle donnera le plus petit des diviseurs qu'elle aura trouvé.

Pour cela, on va adopter un algorithme aussi simple qu'efficace : on va diviser l'entier N par tous les nombres entiers successifs, et voir ce que cela donne.

Encore faut-il tenir compte, dès le départ, du cas particulier que pose 1. On pourra également tout de suite après tester si N = 2, puis éliminer tous les nombres pairs. La parité peut être constatée de la façon suivante : N étant le nombre, on le divise par 2 et on prend la partie entière du résultat qu'on multiplie ensuite par 2. Il est évident que si le nombre est pair, on va retrouver N purement et simplement ; mais si N était impair, le résultat sera différent de N, d'une valeur unitaire.

Par exemple, soit N = 32 ; en divisant par 2, on trouve 16 ; la partie entière de 16 (la partie non décimale) reste 16, que l'on multiplie par 2 et l'on retrouve 32, le nombre d'origine. Mais si N = 37, en le divisant par 2 on trouve 18,5 dont la partie entière vaut 18 ; multipliée par 2, elle donne 36, différente de 37. Par conséquent, si l'on soustrait du nombre N d'origine la valeur trouvée ainsi, on obtient *zéro* si le nombre était pair, et 1 dans le cas inverse.

Après quoi, et puisque les nombres pairs ne sont pas des nombres premiers (à l'exception de 2), on pourra l'afficher. Restent alors les nombres impairs : pour les tester, il suffira de tenter de les diviser par les nombres impairs successifs à partir de 3.

Si l'on trouve un nombre impair D qui divise N, on l'affiche et le programme est terminé. Si aucun nombre impair compris entre 3 et la racine carrée de N ne divise exactement N, on peut affirmer que N est premier. Pour savoir si D est un diviseur de N, la méthode consiste à tester le reste de la division entière de N par D.

Voyons ce programme, qui suit très exactement le processus indiqué :

```
1*RECØNNAITRE SI UN NØMBRE EST PREMIER
5 AFFICHER ' N = ';LIRE N
10 SI N=1 ALØRS DEBUT AFFICHER '1 EST UN CAS PARTICULIER';TERMINER FIN
15 SI N=2 ALØRS ALLER EN 100
25 K←N-ENT(N/2)*2
30 SI K=0 ALØRS DEBUT AFFICHER N,' EST DIVISIBLE PAR 2';TERMINER FIN
35 D←3
37 K←N-ENT(N/D)*D
45 SI K=0 ALØRS DEBUT AFFICHER N,' EST DIVISIBLE PAR ',D;TERMINER FIN
47 SI D>ENT(RAC(N)) ALØRS ALLER EN 100;D←D+2;ALLER EN 37
100 AFFICHER N,' EST PREMIER'
105 TERMINER
```

A la ligne 30, on voit que le test de parité se traduit par K = 0 si elle existe. En cas d'imparité, on charge 3 dans le diviseur D (ligne 35) et l'on divise N par D ; comme dans le cas de la recherche de parité, on prend la partie entière du résultat qu'on multiplie par D. Si la division tombait juste, sans résultat décimal, on retrouverait le N d'origine qui, soustrait de N, donnerait zéro ; par contre, si la division fournit un résultat décimal, la différence entre N et ENT (N/2) * 2 est non nulle et il faudra poursuivre le test.

On le fera en ajoutant successivement des 2 à D pour trouver les diviseurs suivants impairs (ligne 47) puisqu'on a commencé par attribuer la valeur 3 à D, et cela, tant que D ne dépasse pas la racine carrée de N. Ce point est vérifié par la même instruction où apparaît la fonction numérique RAC, pour « racine carrée » de l'expression qui suit.

Lançons ce programme :

```
EXECUTER A PARTIR DE 1

 N = 1
1 EST UN CAS PARTICULIER
TERMINE EN LIGNE 010
  EXECUTER A PARTIR DE 5

 N = 169
169  EST DIVISIBLE PAR 13
TERMINE EN LIGNE 045
```

On a procédé à deux exécutions ici, sur les nombres 1 et 169 ; vérifiez si d'autres nombres sont premiers ou non.

Il existe souvent de très nombreuses façons de résoudre un problème en informatique, et la fantaisie ou l'humeur du programmeur, son art aussi, bien sûr, interviennent pour beaucoup dans la méthode qu'il adoptera.

Par exemple, on aurait tout aussi bien pu rédiger un programme menant au même résultat en faisant appel à une boucle du type FAIRE... TANT QUE. Voici ce que cela donnerait :

```
LISTER A PARTIR DE 1
1*RECØNNAITRE SI UN NØMBRE EST PREMIER
5 AFFICHER ' N = ';LIRE N
10 SI N=1 ALØRS DEBUT AFFICHER '1 EST UN CAS PARTICULIER';TERMINER FIN
15 SI N=2 ALØRS ALLER EN 100
25 K←N-ENT(N/2)*2
30 SI K=0 ALØRS DEBUT AFFICHER N,' EST DIVISIBLE PAR 2';TERMINER FIN
35 FAIRE 50 PØUR D←3 PAS 2 JUSQUA ENT(RAC(N))
40 K←N-ENT(N/D)*D
45 SI K=0 ALØRS DEBUT AFFICHER N,' EST DIVISIBLE PAR ',D;TERMINER FIN
50
100 AFFICHER N,' EST PREMIER'
105 TERMINER
```

On remarquera que la première ligne de ce programme, non numérotée, est une commande donnée à la machine : on frappe réellement LI et le symbole de fin de ligne ; l'ordinateur complète LISTER A PARTIR DE, à la suite de quoi on frappe « 1 ». Cette ligne pourrait, en fait, précéder tous les programmes donnés jusqu'à présent car elle fait afficher le programme courant à partir de la ligne désignée jusqu'à la fin.

Ce n'est qu'à l'instruction de la ligne 35 que les choses changent, par rapport à la version précédente, avec une boucle se terminant à la ligne 50. Cette instruction commande l'affectation de 3 à D, initialement, valeur qu'on accroîtra par pas de 2 jusqu'à la limite, racine carrée de N.

Lancé, un tel programme donnera des résultats semblables et, par exemple :

```
EXECUTER A PARTIR DE 1

 N = 121
121  EST DIVISIBLE PAR 11
TERMINE EN LIGNE 045
EXECUTER A PARTIR DE 1

 N = 23
23  EST PREMIER
TERMINE EN LIGNE 105
```

Rappelons à nouveau que ce qui a été souligné ici correspond à ce que l'opérateur doit frapper au clavier, fins de lignes non compris.

Exercice :

Modifier ce programme pour trouver aussi le second plus petit diviseur, s'il y en a un, des nombres non premiers.

5. LE JEU DE MARIENBAD

Voici un jeu, assez complexe, qui a été programmé par MM. Léger et Lafond, le jeu de Marienbad. Un film célèbre intitulé « L'année dernière à Marienbad », d'Alain Resnais l'a remis en vogue.

Très simple mais mystérieux pour le profane, le jeu de Marienbad se joue à deux avec 16 allumettes disposées en 4 rangées, respectivement de 7, de 5, de 3 et de 1 allumettes. Chaque joueur, à tour de rôle, a le droit de prendre autant d'allumettes qu'il le désire *mais dans une seule rangée uniquement,* à son choix. Celui qui prend la dernière allumette a gagné.

La disposition des allumettes est ainsi la suivante :

```
| | | | | | |
| | | | |
| | |
|
```

Avant de programmer ce jeu et de jouer contre l'ordinateur, essayez quelques parties avec un adversaire de votre entourage. Vous constaterez vite que, pour gagner, la règle à suivre n'est pas évidente.

Pourtant, il existe un algorithme qui permet de se placer dans une situation gagnante, et c'est lui que nous allons programmer. Il exige une conversion binaire et répond au processus suivant :

1) Le nombre d'allumettes de chaque rangée est converti en binaire.

Rappelons les règles de conversion binaire :

Décimal	Binaire
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1 000
etc.	etc.

2) On ajoute les « bits » binaires par colonne, pour les quatre rangées, *sans tenir compte de la retenue.*

Au départ, on a donc :

1re rangée = 7 donne 111
2e rangée = 5 donne 101
3e rangée = 3 donne 11
4e rangée = 1 donne 1

Total binaire par colonne et sans retenue 000

Le total doit précisément être nul pour que la « situation soit gagnante ». Encore faut-il que ce soit votre adversiare qui commence, détruisant forcément ce bel état que vous vous empresserez de rétablir.

Par exemple, si votre adversaire prend 4 allumettes de la première rangée, il reste :

Vous devrez en prendre 4 dans la seconde pour rétablir le jeu suivant :

En transposant les nombres d'allumettes par rangée en binaire et répétant les additions sans retenue, vous constaterez que le total nul a bel et bien été rétabli.

Si vous vous trouvez dans une situation perdante, par exemple, si c'est vous qui commencez à jouer, la seule solution consiste à faire durer la partie en espérant que votre adversaire, ne connaissant pas l'algorithme gagnant, vous permettra de rétablir une situation en votre faveur à un moment ou à un autre.

L'ordinateur appliquera cette stratégie également ; par conséquent et s'il est dans une situation perdante, vous pouvez le faire perdre à la condition de ne pas commettre d'erreur.

Le programme est le suivant :

```
APPELER MABAD
LISTER A PARTIR DE 1
1*SS;LEGER/LAFOND;15/11/74
3*MABAD
5*JEU DE MARIENBAD
10 TABLEAU A[4];A[1]←7;A[2]←5;A[3]←3;A[4]←1
11 AFFICHER[4/,20X,5'*',' JEU DE MARIENBAD ',5'*',3/]
12 AFFICHER['CHAQUE JOUEUR CHOISIT UNE RANGEE ET RETIRE AUTANT ']
13 AFFICHER['D''ALLUMETTES QU''IL',/,'VEUT. CELUI QUI RETIRE ']
14 AFFICHER['LA DERNIERE ALLUMETTE DU JEU A GAGNE.'];&AF(A)
17 CHAINE RE;AFFICHER[3/,'VOULEZ VOUS JOUER LE PREMIER (OUI/NON) ? ']
18 LIRE RE;SI GRL(RE,1)='NON' ALORS DEBUT A[1]←6;ALLER EN 60 FIN
20* JEU DU JOUEUR
21 AFFICHER[2/,'A VOUS DE JOUER...',/,'RANGEE NUMERO : '];LIRE I
22 SI (I-1)*(I-2)*(I-3)*(I-4)≠0 ALORS ALLER EN 25
23 AFFICHER[5X,'NOMBRE D''ALLUMETTES RETIREES : '];LIRE L
24 SI L>0 ET L<=A[I] ALORS ALLER EN 26
25 AFFICHER 'VOUS AVEZ TRICHE!...REJOUEZ...';ALLER EN 21
26 A[I]←A[I]-L;&AF(A)
27 X←0;FAIRE 27 POUR I←1 JUSQUA 4;X←X+A[I]
28 SI X=0 ALORS DEBUT AFFICHER[2/,'VOUS AVEZ GANE'];ALLER EN 31 FIN
29 SI X≠1 ALORS ALLER EN 41
30 AFFICHER['J''AI GAGNE']
31 AFFICHER[2/,'VOULEZ VOUS REJOUER (OUI/NON) ? '];LIRE RE
32 SI GRL(RE,1)='OUI' ALORS ALLER EN 10 SINON TERMINER
40* JEU DE L'ORDINATEUR
41 &PAIR(A,Y);SI Y≠0 ALORS ALLER EN 51
42* L'ORDINATEUR EST DANS UNE SITUATION PERDANTE
43 K←1;FAIRE 44 POUR I←2 JUSQUA 4
44 SI A[I]>A[K] ALORS K←I
45 A[K]←A[K]-1;ALLER EN 60
```

```
50* L'ORDINATEUR EST DANS UNE SITUATION GAGNANTE
51 FAIRE 55 POUR I←1 JUSQUA 4
52 SI A[I]=0 ALORS DEBUT J←I;ALLER EN 55 FIN
53 FAIRE 54 POUR J←1 TANT QUE A[I]>0
54 A[I]←A[I]-1;&PAIR(A,Y);SI Y=0 ALORS ALLER EN 60
55 A[I]←A[I]+J-1
60 AFFICHER[2/,'A MOI DE JOUER...'];&AF(A)
61 SI A[1]+A[2]+A[3]+A[4]=0 ALORS ALLER EN 30 SINON ALLER EN 21
100 PROCEDURE &AF(A) LOCAL A,I;AFFICHER[/]
101 FAIRE 102 POUR I←1 JUSQUA 4
102 AFFICHER[/,10X,F1.0,' ::',5X,*'I ']I,A[I]
103 AFFICHER[/];RETOUR
150 PROCEDURE &PAIR(A,Y) LOCAL A,I,J,X;TABLEAU X[3]
151 FAIRE 154 POUR J←1 JUSQUA 3;X[J]←0
152 FAIRE 153 POUR I←1 JUSQUA 4
153 X[J]←X[J]+A[I]-2*ENT(A[I]/2);A[I]←ENT(A[I]/2)
154 SI X[J]#2*ENT(X[J]/2) ALORS DEBUT Y←1;RETOUR FIN
155 Y←0;RETOUR
```

Dans la ligne 100, la procédure &AF fait afficher les rangées d'allumettes restantes. A la ligne 150, la procédure &PAIR effectue le traitement suivant : transformation du nombre d'allumettes de chaque rangée en base 2 ; sommation des nombres convertis pour chaque puissance de 2 ; cette sommation étant exécutée en base 10, on teste si la somme est paire ($y = 0$) ; si elle est impaire, y est mis à un.

Le programme lancé, on pourra jouer des parties telles que les suivantes :

```
EXECUTER A PARTIR DE 1


                    ***** JEU DE MARIENBAD *****

CHAQUE JOUEUR CHOISIT UNE RANGEE ET RETIRE AUTANT D'ALLUMETTES QU'IL
VEUT. CELUI QUI RETIRE LA DERNIERE ALLUMETTE DU JEU A GAGNE.

          1 ::      I I I I I I I
          2 ::      I I I I I
          3 ::      I I I
          4 ::      I


VOULEZ VOUS JOUER LE PREMIER (OUI/NON) ? OUI

A VOUS DE JOUER...
RANGEE NUMERO : 1     NOMBRE D'ALLUMETTES RETIREES : 5

          1 ::      I I
          2 ::      I I I I I
          3 ::      I I I
          4 ::      I
```

```
A MOI DE JOUER...

          1 ::      I I
          2 ::
          3 ::      I I I
          4 ::      I


A VOUS DE JOUER...
RANGEE NUMERO : 3       NOMBRE D'ALLUMETTES RETIREES : 4
VOUS AVEZ TRICHE!...REJOUEZ...

A VOUS DE JOUER...
RANGEE NUMERO : 3       NOMBRE D'ALLUMETTES RETIREES : 2

          1 ::      I I
          2 ::
          3 ::      I
          4 ::      I


A MOI DE JOUER...

          1 ::
          2 ::
          3 ::      I
          4 ::      I


A VOUS DE JOUER...
RANGEE NUMERO : 3       NOMBRE D'ALLUMETTES RETIREES : 1

          1 ::
          2 ::
          3 ::
          4 ::      I
J'AI GAGNE

VOULEZ VOUS REJOUER (OUI/NON) ? NON
TERMINE
```

On constatera que le programme a même prévu que le joueur puisse tenter de tricher... Par contre, ce défaut est inconnu de la machine (qui, en toute rigueur, ne pourrait commettre qu'une « erreur »).

Exercice :

Que faudrait-il modifier dans ce programme pour inverser la règle donnant le gagnant, c'est-à-dire pour faire que celui qui prend *la dernière allumette* perd *la partie ?*

Chapitre XV

MAINTENANCE ET FIABILITE

1. INTRODUCTION A LA NOTION DE MAINTENANCE

Un exemple pris dans la vie courante va nous permettre de bien cerner cette notion de maintenance.

Supposons l'achat d'un téléviseur ; de manière courante, le produit acheté est garanti durant une année. Cette garantie, dite *garantie usine,* permet à l'acheteur de disposer, sans coût supplémentaire, car elle est incluse naturellement dans le prix de vente, du matériel pendant sa durée de vie normale.

• *Disposer du matériel* signifie que celui-ci est utilisable, de manière normale, dans des conditions normales de fonctionnement.

• *Durée de vie normale :* c'est là une notion subjective, difficile à cerner et très spécifique du type de matériel concerné ; elle est de l'ordre de quelques années, de cinq à dix.

Nous reviendrons ultérieurement sur ces notions de garantie, de durée de vie. Elles sont de toute façon incluses dans la définition de la maintenance.

On appelle *maintenance* l'ensemble des actions qui visent à entretenir un matériel de manière que celui-ci offre à son utilisateur une qualité de service suffisante.

Le service, tel est le mot-clé. Comme on le croit trop souvent, la maintenance n'est pas un ajout à la vente, un prolongement de celle-ci. Elle n'est pas un service complémentaire offert au gré du vendeur, mais une partie indissociable de la vente. Sa non-exécution peut aller jusqu'à la mise en cause de la vente.

1.1. La garantie

On peut distinguer deux types de garanties : la garantie *usine* et la garantie *en état de bon fonctionnement.*

• *La garantie usine :* appelée aussi garantie de fabrication, elle s'applique dans tous les cas de vice de construction ou de conception.

Naturellement incluse dans le prix de vente du matériel, la garantie usine a pour but de permettre à l'acheteur de disposer, pour le ou les modes de fonctionnement prévus de manière classique, d'un matériel en bon état de fonctionnement pendant sa durée de vie normale.

• *La garantie en bon état de fonctionnement :*

Elle diffère de la garantie usine de deux manières en général :

— *la durée :* elle est de l'ordre d'une année dans le cas de la garantie usine ;

— *le lieu et le mode d'exécution :* si la garantie usine met généralement en œuvre un pur remplacement de la partie défectueuse, la garantie en bon état de fonctionnement ne fait pas appel à la notion d'*échange* mais plutôt à celle de *réparation.*

1.2. La maintenance en informatique

Complément indispensable de la vente, la maintenance informatique comprend deux étapes :

• Elle recouvre tout d'abord l'installation du matériel et la formation des acheteurs.

• Elle englobe ensuite un véritable soutien proposé par le constructeur à ses clients, soutien à caractère *préventif* et *curatif.* Ce soutien est assuré par une structure spécifique dépendant du constructeur, l'*Après-Vente* ou *Service d'entretien et de maintenance,* généralement de type décentralisé pour répondre aux exigences de la clientèle en matière de temps d'intervention.

1.3. Installation

Si les produits grand public comme le téléviseur ou la Hi-Fi ne donnent parfois pas lieu à installation, il n'en est pas de même en électronique ou en informatique où il s'agit d'une règle classique.

Installer le matériel, c'est le livrer à son lieu de destination finale, le monter si plusieurs pièces ou sous-ensembles le constituent, effectuer les branchements nécessaires, le mettre sous tension et réaliser le ou les réglages éventuellement nécessaires. A l'issue de l'installation, le matériel est réputé utilisable.

Dans le domaine des produits grand public, cette dernière notion est simple à mettre en œuvre. Rendre un téléviseur utilisable, c'est effectuer tous les branchements nécessaires (secteur, antenne), réaliser un certain nombre de réglages (chaque chaîne a son canal, correspond à une fréquence), s'assurer que la réception de l'image et du son est correcte.

En informatique, la situation est tout autre : **le système informatique ne prendra toute sa véritable valeur que dans le cadre d'une application. Deux cas peuvent alors se produire :**

— L'installation et la vérification du bon état de fonctionnement sont réalisés sur des programmes tests fournis par le constructeur lui-même. Ces programmes sont, de conception, suffisamment généraux pour pouvoir tester les principaux composants du système — unité et mémoire centrale, mémoire auxiliaire, terminaux et périphériques.

— L'installation est réalisée par le constructeur ; la vérification du bon état de fonctionnement est réalisée sur une famille de programmes ayant donné lieu à concertation entre le fournisseur et l'acheteur — programmes fournis par le constructeur comme dans le cas précédent, ainsi que par l'acheteur.

Dans ce dernier cas, il s'agira d'applications types exécutées en vue de vérifier, non seulement le bon fonctionnement dans son ensemble, mais également certaines particularités de la configuration.

Dans certains cas, prévus contractuellement, c'est l'application ou tout un ensemble d'applications du client qui sont testés lors de la procédure d'installation du système informatique.

L'ensemble des opérations constitue ce que l'on appelle la *réception* ou la *recette* du système. La recette terminée, un procès-verbal est généralement signé entre le fournisseur et l'acheteur.

La date de signature du procès-verbal de recette, fixe la date de départ de la garantie de bon fonctionnement.

On notera que les ventes O.E.M. *(Original Equipment Manufacturer)* constituent une exception à tout ce qui vient d'être dit. Les matériels vendus selon la formule O.E.M. donnent, certes, lieu à une procédure de recette lors de leur livraison à l'acheteur. Ce dernier, cependant, les a achetés en vue de les intégrer à des configurations plus complexes, afin d'être utilisées dans des conditions non maîtrisées, voire même inconnues du fournisseur O.E.M. Dans ces conditions, l'acheteur final a généralement tendance à se tourner, pour ce qui est de l'exercice normal de la garantie, vers *l'intégrateur* et non pas vers le fournisseur O.E.M.

Il est bien évident que fournisseur O.E.M. et intégrateur ont entre eux des relations contractuelles bien précises ; celles-ci sont cependant tout à fait étrangères à l'utilisateur final.

1.4. Matériel et logiciel

Sans logiciel, un matériel est inerte, inutilisable, de même qu'un marteau pris isolément n'a aucun sens. Son utilité n'apparaît en effet que lorsque un manipulateur l'utilise avec un clou ; un ordinateur ne prend vie que sur une application bien déterminée.

Le logiciel lui-même recouvre en réalité deux grandes familles de programmes :

• *Les programmes de base :* fournis par le constructeur, ils ont pour noms système d'exploitation, traducteurs de langages, utilitaires, ...

> *Exemples :*
>
> • *Le BDOS, le BASIC et le LSE pour le micro-ordinateur LX 500*

Ils ont pour but d'offrir à l'utilisateur les services de base indispensables : chargement de programme, gestion des fichiers, gestion des périphériques, systèmes de programmation, gestion de la mémoire, mise en œuvre des ressources diverses...

• *Les programmes d'application :*

Parfois fournis par les fournisseurs de matériels — il s'agit alors de programmes standard dits progiciels d'application — ce sont généralement les programmes rédigés par les utilisateurs eux-mêmes.

> *Exemples :*
>
> • Progiciels d'application :
> *comptabilité générale, analytique, gestion de trésorerie, gestion de stocks...*
>
> • Applications :
> *traitement de transactions financières au guichet, réservation de places, émission de billets, demande de renseignements...*

Dans le cas des systèmes complets — matériel et logiciel —, on conçoit ainsi que la maintenance est une activité fort complexe qui implique que soient mises en œuvre, aux plans contractuels et techniques, des procédures fixant les droits et les devoirs des différentes parties.

2. LE CONTRAT DE MAINTENANCE

Celui-ci s'exerce aussi bien durant, qu'après la période de garantie. Après, il est bien entendu payant, le coût reflétant directement l'étendue des services offerts.

2.1. La location

La pratique de la location, assez peu courante en électronique professionnelle, l'est beaucoup plus en informatique du fait de son introduction, pour des raisons économiques et financières, par les constructeurs de grande informatique, il y a de cela quelques années.

L'évolution vers une informatique plus légère, décentralisée, à base de mini ou micro-ordinateurs a vu cette pratique se généraliser et aujourd'hui tous les constructeurs la proposent à leur clientèle.

Au plan de la maintenance, la notion de location n'introduit aucune différence. L'utilisateur qui loue un matériel est contractuellement lié au fournisseur, le contrat recouvrant :

— la mise à disposition,
— la maintenance,

du matériel.

2.2. La maintenance au forfait

Le forfait garantit à l'utilisateur un service complet, sur une durée déterminée, et ce pour un prix fixé à l'avance.

Le service a trait à la disponibilité du matériel concerné ; diverses clauses contractuelles précisent généralement :

— de quelle manière et dans quel délai les interventions seront assurées ;
— sur quels ensembles, sous-ensembles, pièces, la garantie et le forfait s'exercent, et par voie de conséquence, quels éléments en sont exclus.

Au plan financier, le forfait maintenance est révisable en fonction des autorisations gouvernementales.

Les exclusions peuvent donner lieu à des avenants au contrat.

La formule est souple et offre à l'utilisateur une bonne sécurité de fonctionnement. Souvent conseillée par les constructeurs, elle peut être assimilée à une « assurance de fonctionnement ».

2.3. La maintenance au coup par coup

Dans une telle formule, les interventions de réparation ou d'entretien sont réalisées et facturées au coup par coup. A priori, cette formule peut paraître moins chère et plus souple ; c'est loin d'être le cas, car si elle est mise en œuvre sur demande de l'utilisateur (soit lors d'un incident, soit pour une visite préventive), elle n'entre pas nécessairement à l'instant où la demande est émise, dans le planning du fournisseur. En d'autres termes, la qualité de service est généralement inférieure, non pas au plan technique, mais du fait d'éventuels délais d'intervention.

2.4. Le contrat d'entretien

Dans ce cas, les interventions sont limitées au seul entretien, les réparations proprement dites donnant lieu à facturation au coup par coup.

2.5. Paramètres à prendre en compte dans un contrat de maintenance

Le contrat de maintenance a, on l'a vu, pour but de garantir à l'utilisateur un service de bon fonctionnement. Liant contractuellement utilisateur et fournisseur, il précise les droits et devoirs de chacun. Pour ce qui est du *fournisseur,* celui-ci s'engage à installer un matériel conforme aux désirs de l'utilisateur, en bon état de fonctionnement. Type et durée de la garantie sont expressément mentionnés, de même que le type et le prix des services associés à l'entretien (préventif ou curatif) et au dépannage.

Si *l'utilisateur,* c'est-à-dire l'acheteur, a des droits, il a aussi des devoirs. Le matériel qui lui est livré ne l'est que dans un cadre précis d'utilisation. La température ambiante, la tension des différentes sources d'alimentation, la fréquence et la nature de certaines opérations à caractère préventif, le ou les modes d'exploitation possibles, constituent quelques-uns des points auxquels l'utilisateur doit être attentif. L'ensemble des précautions à prendre, des procédures à mettre en œuvre, des contrôles à faire figure généralement dans le contrat ou dans des annexes à celui-ci. En ne les respectant pas, volontairement ou non, l'utilisateur peut se mettre hors des conditions normales d'application du contrat.

Droits et devoirs respectifs sont à préciser au moment de la négociation du contrat de vente ou de location.

L'on portera une attention particulière aux points suivants :

- conditions d'installation ;
- durée et étendue de la garantie ;
- conditions de renouvellement de la garantie ;
- étendue, prix, délai d'intervention de la maintenance ;
- taux horaire, prix des pièces, coût moyen d'une réparation ;
- nécessité parfois d'un devis préalable ;
- facturation séparée ou non des déplacements, pièces, frais d'expédition...

3. MAINTENANCE PREVENTIVE-CURATIVE

Si le mode d'intervention est le même, l'objectif constant étant de conserver au système son intégrité, le moment auquel se situe cette intervention peut varier.

3.1. Maintenance curative

Après un incident, l'action de maintenance a pour objet de dépanner le système après avoir identifié la cause première de l'incident. Le technicien de maintenance dans ce cas intervient pour remettre en état, *c'est la maintenance curative.*

Ce type d'intervention, bien entendu nécessaire, présente cependant bien des inconvénients ; en particulier, l'intervention après l'incident peut être de durée variable. Il faut, en effet, essayer de recréer les conditions de l'incident, en trouver les causes profondes, agir en sorte que ces causes ne se reproduisent plus, vérifier que des effets secondaires ne se sont pas manifestés, etc.

Cette intervention à chaud est donc des plus inconfortables, pour tous les acteurs impliqués par le système. En informatique, comme dans le domaine de la santé, le dicton « mieux vaut prévenir que guérir » prend toute sa valeur. C'est là l'objet de la *maintenance préventive.*

3.2. Maintenance préventive

Dans ce type de maintenance, l'on agit de manière à détecter une défaillance avant que celle-ci ne se manifeste et n'altère les résultats d'un traitement quelconque.

L'action du technicien de maintenance va ainsi s'exercer sur les différents sous-ensembles du système, une attention particulière étant portée sur ceux dont la probabilité de panne est la plus forte. Elle recourt en particulier à des mesures de valeurs nominales, des réglages, des actions de nettoyage, des vérifications de soudures, de contacts, des mesures de température, de tolérances, ... et met en œuvre comme moyens : des programmes de tests d'ensemble, des programmes spécifiques aux sous-ensembles, des instruments de mesure et des testeurs de cartes.

4. MAINTENANCE ET FIABILITE

La maintenance préventive a, on vient de le voir, pour objet de détecter, avant qu'il ne se produise, tout risque d'incident. Ce risque fait intervenir la notion de *fiabilité* des diverses entités composant le système.

4.1. Fiabilité d'un élément

Traduction française du terme américain « reliability », la fiabilité d'un élément caractérise la durée de vie et le comportement de cet élément.

La durée de vie

Conçu pour réaliser une ou plusieurs fonctions bien précises, un élément électronique est prévu pour fonctionner dans des conditions d'utilisation (tensions, valeurs de courants, température, degré hygrométrique de l'atmosphère ambiante) bien déterminées.

Pour un élément mécanique, il en est de même ; les conditions d'utilisation et les paramètres de l'environnement influent sur l'usure, les déformations de cet élément.

Le comportement

Soumis à diverses contraintes (tensions, vitesse de rotation, etc.), l'élément doit, pour des configurations d'entrée bien précises, fournir des réponses non moins précises.

Tout écart entre la réponse obtenue et la réponse attendue traduit un phénomène de mauvais fonctionnement. C'est un clignotant ; la panne est prévisible.

4.2. Fiabilité du système

L'assemblage de tous les éléments d'un système conduit à la notion de fiabilité du système. Si λ_n caractérise le taux de défaillance du composant *n*, le taux de défaillance du système λ est alors donnée par :

$$\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_n$$

Ce taux de défaillance a les dimensions d'une probabilité par rapport au temps, et s'exprime souvent en pourcent de risques par 100 heures. Ainsi, un pourcent par 1000 heures donne un taux de défaillance de 10^{-5}. Plus ce taux est petit, plus la fiabilité du composant ou du système est grande.

Une panne au niveau d'un composant a ainsi une incidence directe sur le système.

Pour améliorer la fiabilité d'ensemble, on peut mettre des composants en parallèle, soit introduire une certaine redondance. Dans ces conditions, on a :

$$\frac{1}{\lambda} = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} = \frac{\lambda_1 + \lambda_2}{\lambda_1 \cdot \lambda_2}$$

soit :

$$\lambda = \frac{\lambda_1 \cdot \lambda_2}{\lambda_1 + \lambda_2}$$

4.3. Le Temps moyen entre chaque panne ou MTBF

Le M.T.B.F. *(Medium Time Between Failure)* ou *temps moyen entre chaque panne* caractérise la fiabilité d'ensemble d'un système. S'exprimant en heures, défini par le constructeur, il fournit à l'utilisateur une indication sur la durée de temps durant laquelle il est peu probable qu'un incident se manifeste.

Le M.T.B.F. fournit aussi une indication à prendre en compte dans la mise en œuvre des actions de maintenance préventive.

Le M.T.B.F. est défini par le constructeur, sur la base des probabilités de vie des divers éléments constituant le sous-ensemble ou l'ensemble du système. Sa détermination et entachée cependant d'une certaine subjectivité car le M.T.B.F. est lié directement à des conditions d'utilisation très précises.

Exemples de M.T.B.F. :

1. Imprimante LX 80 :
— *composants mécaniques : 2 400 heures*
— *composants électroniques : 4 800 heures*

2. Lecteur de disquettes LX 45 D :
5 000 heures

Exemples de conditions d'emploi :

1. LX 80 :

• Conditions normales d'utilisation :
— *8 heures/jour ;*
— *impression de 600 000 caractères/jour.*
• Durée de vie :
— *de la tête d'impression : 300 millions de caractères ;*
— *des autres composants : 32 millions de lignes de longueur moyenne ou 8 années d'utilisation normale ;*
— *du ruban : 2 millions de caractères.*
• Périodicité des interventions de maintenance :
— *tête d'impression : 100 millions de caractères soit 8 mois d'utilisation dans des conditions normales ;*
— *autres composants : 1 million de lignes de longueur moyenne soit 3 mois d'utilisation dans des conditions normales.*

2. Lecteur de disquettes

• Conditions d'environnement :
— *température : + 10 °C à + 45 °C ;*
— *humidité : + 20 % à + 80 %.*
• Spécifications électriques :
— *entraînement du disque ;*

...

— source de tension :
220 V (+ 10 %, − 15 %) à 50 Hz
115 V (+ 10 %, − 15 %) à 60 Hz
— attelage tête de lecture/écriture :
42 V ± 2 V
— électronique :
trois possibilités :
± 12 V et + 5 V
+ 24 V et ± 5 V
+ 24 V et + 5 V
et − 12 V à − 15 V.

4. Le temps moyen de réparation ou M.T.T.R.

Le M.T.T.R. *(Mean Time to Repair)* ou *temps moyen de réparation* s'exprime en heures ou en minutes ; plus le temps est faible, plus la réparation est aisée. Le M.T.T.R. caractérise ainsi la facilité avec laquelle la réparation, la maintenance peuvent être effectuées ; cette facilité est recouverte par la notion de *maintenabilité.*

Cette définition, volontairement caricaturale, recouvre en réalité des procédures de contrôle, de détection et de réparation souvent complexes.

Le M.T.T.R. est aujourd'hui considéré comme étant la somme de différents temps :

— le temps de *réparation ;*
— le temps de *dépannage ;*
— le temps de *voyage ;*

De plus en plus, afin de diminuer le temps d'*indisponibilité* du système des techniques d'*échange standard* sont appliquées. Un sous-ensemble défectueux est ainsi totalement désolidarisé du système, remplacé par un sous-ensemble équivalent en bon état de fonctionnement ; on garantit ainsi une bonne disponibilité au système dans son ensemble.

Le sous-ensemble défectueux quant à lui sera ultérieurement :

— soit réparé sur place ;
— soit réparé en usine, en atelier.

Exemple de M.T.T.R.

Imprimante LX 80 : *30 minutes.*

ANNEXES

1. LE LANGAGE BASIC AU COMPLET

1.1. Les instructions

DIM
REM
[LET] variable = expression *(Les crochets indiquent que le contenu est éventuel.)*
LP [n;] L
PRINT [n;] [expression ou fonction TAB] ... [;expression ou fonction TAB] [;]
PR
TAB (expression numérique) (fonction)
INPUT [n;] ["constante";] variable [;variable...] [;"constante";variable] [;...]
READ variable [;variable...]
DATA constante [;constante ; constante.....]
RESTORE numéro de ligne
INPTC [n;] ["constante";] variable [;variable...] [;"constante";variable] [;...]

1.2. Instructions de branchement

GOTO Numéro de ligne
IF...THEN...ELSE

	[GOTO] n		[GOTO] m
IF condition THEN	instruction	ELSE	instruction

FOR variable numérique = expression TO expression [STEP expression num]
NEXT [Variable]
EXIT numéro de ligne
GOSUB numéro de ligne
ON Variable numérique GOTO a;b;c...
END
STOP
LINK " nom du fichier "
POKE adresse mémoire ; valeur

1.3. Les fichiers

OPEN n;"nom de fichier" [;variable numérique]
CLOSE n
READ n; variable [;variable].
WRITE n; variable [;variable...]
TYP (n) **(fonction)**
GET n;p; variable [;variable...]
PUT n;p; variable [;variable...] [;NOEOF]

1.4. Les fonctions

SQR (expression numérique)
SIN (expression numérique)
COS (expression numérique)
LOG (expression numérique)
EXP (expression numérique)
RND (expression numérique)
SGN (expression numérique)
ABS (expression numérique)
INT (expression numérique)
LEN (expression alphanumérique)
VAL (chaîne)
STR$ (expression numérique)
ASC (variable alphanumérique)
CHR$ (variable numérique)
PEEK (adresse mémoire)
CALL (adresse mémoire ; valeur décimale)

2. L'INTERPRETEUR, L'EDITEUR ET LE BDOS

LES COMMANDES DE L'INTERPRETEUR BASIC

LOAD nom de fichier [;U] *(U : unité contenant le fichier (A ou B).)*
SAVE nom de fichier [;U]
RUN [numéro de ligne]
CLEAR
NUMB [n[;p]]
LIST [n;] [p [;q]]
QUIT
RESUME
APPEND nom de fichier [;U]
EDIT
DEL n1;n2
WAIT p

EDITEUR DE TEXTE

EDIT n

LES COMMANDES DE BDOS

B [;nom [; code unité]]
C; nom; L; type [;code unité]
S; nom [;code unité]
L [; numéro de canal] [;code unité]
I; code unité *(obligatoire)*
R
F; nom1; nom2; code unité
F; nom1; nom2
D; A; B
D; A; A
D; A

3. PETIT VOCABULAIRE DU TRAITEMENT DE L'INFORMATION

A

ACCES : - *accès direct :* technique permettant l'accès immédiat aux données, ceci indépendamment de l'ordre dans lequel celles-ci sont rangées en mémoire, et quel que soit l'ordre de la demande. - *accès séquentiel :* il s'agit d'un mode d'accès consistant en la lecture ou l'écriture de données dans l'ordre où celles-ci ont été stockées ou lues. - *temps d'accès :* intervalle de temps entre la demande de mise à disposition de données et la disponibilité de ces mêmes données dans la zone réceptrice.
ACCUMULATEUR : Registre pour stocker le résultat d'une opération arithmétique ou logique ; un registre est ainsi une zone mémoire particulière. Cette zone peut également servir pour toute manipulation ultérieure de données.
ADRESSE : Symbole ou nombre caractérisant un emplacement mémoire. Associé à une information, celui-ci permet de l'enregistrer ou de la retrouver.
ALGOL : Abréviation de : Algorithmic Language ; langage de programmation évolué utilisé dans les disciplines scientifiques telles que la physique, les mathématiques.
ALGORITHME : Ensemble des règles permettant, via la description d'un certain nombre défini d'opérations, de décrire un problème et d'aboutir à sa solution ou à l'obtention d'un résultat.
ALPHANUMERIQUE : Terme caractérisant une donnée pouvant contenir des chiffres, des lettres ou des caractères spéciaux, l'une quelconque de ces familles pouvant être présente seule ou combinée aux autres.

APPLICATION : Ensemble des descriptions logiques permettant, après traitement, d'aboutir à résoudre une ou plusieurs fonctions. La facturation, la paie, la comptabilité sont les applications les plus couramment implantées sur les systèmes destinés aux entreprises. Selon sa complexité, une application peut comprendre une ou plusieurs unités ou chaînes de traitement, correspondant à une ou plusieurs fonctions.
ARITHMETIQUE : Ensemble des opérations basées sur l'application des règles mathématiques ; l'arithmétique s'appuie sur les notions de base, de chiffres et obéit aux règles concernant les opérateurs arithmétiques (+, −, ×, /), et les signes.
ASSEMBLEUR : L'assembleur, ou programme d'assemblage, effectue, par traduction, le passage d'un langage bas niveau à un langage de type machine.

B

BANDE : (magnétique : voir dérouleur)
BATCH : Nom anglo-saxon caractérisant le traitement des données par lots. Les données à traiter sont ainsi tout d'abord groupées. Ce type de traitement s'oppose au traitement immédiat (temps réel) ou au traitement dans les systèmes à priorité.
BAUD : Unité de vitesse de transmission. Se confond avec le nombre de bits transmis par seconde dans le seul cas où la transmission se fait sur deux niveaux de tension.
BCD ou DCB : Le décimal codé binaire (Binary Coded Decimal) est un code dans lequel chaque chiffre décimal est séparément converti en binaire.
BINAIRE : Terme qualifiant le système de numération à base 2 ; basé sur l'utilisation de deux chiffres seulement, le 0 et le 1.
BIT ou BINARY DIGIT : Chiffre binaire, bit, position binaire. De façon générale, ce terme qualifie l'unité élémentaire d'information.
BIT DE PARITE : Elément de contrôle consistant à ajouter à un groupe d'éléments binaires un chiffre binaire de manière que le nombre total des chiffres binaires soit pair (ou impair).
BUFFER ou MEMOIRE TAMPON : Zone mémoire destinée au stockage temporaire de données. La mémoire tampon est utilisée lorsque le transfert des données est réalisé entre dispositifs présentant des modes de fonctionnement différents(mode de transfert, vitesse de transfert...)
BUS ou LIGNE BUS : Connexion multiple permettant la distribution de signaux à des groupes différents de circuits.
BYTE ou MULTIPLET : Groupe de positions binaires considéré comme un tout. Un hexet est un multiplet de 6 bits et un octet un multiplet de 8 bits.

C

CALCUL en VIRGULE FIXE : Système de notation des nombres dans lequel la virgule décimale occupe un emplacement fixe dans la zone mémoire réservée à chaque nombre.

CALCUL en VIRGULE FLOTTANTE : Système de notation des nombres dans lequel l'emplacement de la virgule est variable de manière que l'on dispose toujours du même nombre de chiffres significatifs dans la zone mémoire, réservée à chaque nombre.
CALCULATEUR : Expression qualifiant tout dispositif capable d'exécuter des opérations arithmétiques ou logiques sur des données digitales. Si le programme est figé, inamovible, la machine de traitement de l'information s'appelle calculateur ou calculatrice. Si le programme peut être changé, modifié au gré de l'utilisateur, on parle plutôt d'ordinateur.
CANAL : Organe permettant d'établir une liaison d'échange entre l'unité de calcul et les différentes unités d'entrée/sortie.
CAPACITE (de mémoire) : Quantité d'informations pouvant être enregistrées par une mémoire, souvent mesurée en K (un K correspond à 1 024 positions).
CARACTERE : Expression codée intervenant dans la représentation ou la constitution d'une donnée (lettre, chiffre, signe, symbole) sur un nombre déterminé de positions binaires.
CHIP : Composant unitaire : un chip ou une puce réalise une ou plusieurs fonctions. Terme rendu courant depuis les progrès accomplis dans le domaine de la micro-électronique.
COBOL : Abréviation caractérisant le langage le plus répandu dans la description et le traitement des applications administratives de gestion (Common Business Oriented Language).
CODE : Ensemble des conventions et règles permettant, soit de représenter l'information, soit de passer d'un système de représentation à un autre. Exemple : ASCII : code normalisé américain (American Standard Code for Information Interchange).
CODE D'INSTRUCTION : Symbole ou valeur désignant une instruction dans un langage de programmation. L'ensemble des codes d'instructions caractérise un language.
CODE OPERATION : Symbole ou valeur caractérisant le type d'opération d'une instruction précise.
COMPILATEUR : Programme utilisé pour traduire un programme écrit en langue « source », orienté problème, en un programme exprimé en langage machine ou en langage d'assemblage.
COMPLEMENT : Le complément à n est le nombre qu'il faut ajouter à celui que l'on désire complémenter pour en porter tous les chiffres à la valeur n.
COMPOSANT : Elément de base de la technologie électronique ou informatique. Les transistors, diodes, résistances... constituent des composants discrets. La technologie micro-électronique permet, grâce à l'intégration, de réaliser plusieurs fonctions, voire de constituer de véritables unités centrales sur un seul composant. Les circuits sont dits intégrés.
CONVERSATIONNEL : Mode d'utilisation d'un système de traitement de l'information par lequel utilisateur et système échangent données et informations au travers d'un véritable dialogue.

D

DENSITE : Nombre d'éléments (binaires par exemple) continus par unité de longueur, de surface ou de volume.
DEROULEUR (de bande, de cassette) : Dispositif de lecture et d'écriture sur support magnétique.
DIRECT : voir : Accès.
DISQUE : Dispositif de stockage basé sur les propriétés magnétiques (conservation d'informations) de matières ferromagnétiques ; ces propriétés sont mises en œuvre au travers de supports analogues aux disques.
DISPLAY : Afficheur.
DUPLEX : Mode de transmission entre deux points, via une liaison télégraphique ou téléphonique. - *Semi-duplex ou bi-directionnel à l'alternat :* Les messages sont transmis alternativement dans un sens puis dans l'autre. - *Duplex ou bi-directionnel simultané :* Les messages sont transmis simultanément dans les deux sens.
DUPLIQUER : Reproduire.

E

EDITER : Disposer l'information en groupes ou blocs, avec insertion d'espaces, suppression des zéros à gauche, etc., avant d'en assurer la présentation, sur une imprimante par exemple.
EXPOSANT : L'exposant ou caractéristique est une quantité algébrique accompagnant les chiffres significatifs d'un nombre et représentant la puissance de la base. Sa combinaison avec la mantisse permet d'obtenir la valeur du nombre.
EXTERNE : Adjectif permettant en particulier, de qualifier une mémoire (autre que la mémoire principale), figurant dans la configuration d'un système. Cette mémoire est également dite auxiliaire ou secondaire.

F

FERRITE : Matière possédant la particularité de pouvoir stocker, conserver et restituer l'information. Celle-ci s'appuie sur les propriétés magnétiques de la matière. Utilisées depuis de longues années pour constituer les mémoires centrales d'ordinateurs, les ferrites sont, de nos jours, progressivement remplacées par les mémoires à circuits intégrés.
FICHIER : Ensemble d'informations classées selon un indicatif. Groupés sur un support, (carte, bande magnétique, disquette), celles-ci sont susceptibles d'être accessibles lors d'un même traitement.
FIRMWARE : Programme enregistré dans une mémoire inaltérable (de type ROM par exemple) et accessible à l'ordinateur.
FORTRAN : Abréviation de « Formula Translation » ; langage de programmation principalement utilisé dans l'expression et le traitement d'applications scientifiques.
FORMAT : Caractérise la forme, la disposition d'une donnée. Par extension, s'applique à l'organisation d'un fichier.

H

HARDWARE = Matériel : Terme regroupant l'ensemble des constituants matériels d'un système.
HEXADECIMAL : Système de numération à base 16.
HORLOGE : Dispositif émettant des signaux périodiques de façon à réguler le fonctionnement d'un ordinateur.

I

IDENTIFICATEUR : Nom symbolique permettant de désigner des données dans un langage de programmation.
IMPRIMANTE : Dispositif permettant de restituer, sous forme écrite et selon une présentation précise (voir Format) les informations introduites dans un ordinateur et les résultats de traitements. Selon les modèles, l'imprimante édite les informations caractère par caractère (imprimante caractère), ligne à ligne (imprimante ligne), voire par pages (imprimante pages).
INFORMATION : Ensemble de faits, de notions ou d'instructions susceptibles d'être communiqués, interprétés et traités. Signification attribuée à des données en fonction des conventions utilisées pour les représenter.
INFORMATIQUE : Science du traitement rationnel, notamment par machines automatiques, de l'information considérée comme le support des connaissances et des communications dans les domaines technique, économique et social (définition de l'Académie Française).
INCREMENT : Valeur constante s'ajoutant à une donnée, périodiquement ou non.
INCREMENTER : Augmenter, faire progresser.
INDEX : Repère ; introduit dans un registre, il permet d'effectuer des comptages, des progressions, des itérations.
INFORMATION : Quantité élémentaire, numérique ou alphabétique traitée par l'ordinateur.
INSTRUCTION : Terme désignant une expression décrivant une opération à effectuer. Présenté à la machine, il provoque son exécution.
INTERFACE : Frontière conventionnelle entre deux systèmes, deux dispositifs... permettant les échanges entre ceux-ci, selon des règles bien déterminées.
INTERNE : Qualifie la mémoire principale de l'ordinateur.
INTERPRETEUR : Programme de traduction.
INTERRUPTION : Arrêt dans le déroulement d'un programme suivi du lancement et de l'exécution d'un autre programme. Lorsque l'exécution de ce dernier est terminée, il y a reprise du programme interrompu.

K

KILO = Unité = mille ou 10^3 en décimal, 2^{10} ou 1 024 en binaire. Exemples : un kilo-octets vaut 1 024 octets ; un kilobaud correspond à un débit de 1 000 bauds.

L

LANGAGE : Rassemble, en programmation, l'ensemble des caractères et symboles ainsi que les règles définissant la manière de les combiner pour s'exprimer et se faire comprendre d'un ordinateur. BASIC, COBOL, l'Assembleur, LSE sont des langages.
LANGAGE SOURCE ou LANGAGE ORIGINE : Langage dans lequel s'exprime le programmeur pour traduire un ou plusieurs algorithmes.
LANGAGE SYMBOLIQUE : Terme générique regroupant tout langage de programmation autre que le langage machine et utilisant des symboles.
LANGAGE OBJET : Résultat de la transcription d'un langage symbolique par un programme de transformation (assembleur, compilateur...)
LECTEUR : Dispositif périphérique servant à introduire en mémoire centrale des informations figurant sur des supports externes (disque, carte, bande...)
LOGICIEL : Equivalent français du mot anglo-saxon « SOFTWARE ». Caractérise l'ensemble des programmes qui, appliqués à un matériel (hardware) bien déterminé, permet d'utiliser celui-ci. On distingue le logiciel de base, généralement fourni par le constructeur (voir Système d'exploitation) et le logiciel d'application (voir Application).

M

MAINTENANCE : Ensemble des actions destinées à assurer le bon fonctionnement des constituants d'un système. L'entretien et la mise à jour (préventif ou curatif) sont deux composants essentiels de la maintenance.
MANTISSE : Partie significative d'un nombre exprimé en virgule flottante.
MATRICE : Voir Imprimante.
MEDIUM (pluriel MEDIA) : Support destiné à recevoir, sous forme codée, de l'information. Exemples : bande, disquette magnétique.
MEMOIRE : Dispositif capable de stocker, de conserver et de restituer des données.
MODULAIRE : Permet de qualifier un ensemble qui, s'appuyant sur un noyau de base, peut croître par éléments autonomes. Exemple : système modulaire, programme modulaire...
MODULE : Désigne l'un des éléments autonomes. Possédant tous les moyens permettant de remplir une ou plusieurs fonctions précises, le module peut être intégré à un ensemble d'autres modules pour constituer un système modulaire.
MONITEUR : Programme de supervision et d'enchaînement des opérations réalisées par un ordinateur.
MICROPROGRAMMATION : Action consistant en la programmation des actions destinées à figurer dans des mémoires inaltérables (voir Firmware).
MOT : Groupe de caractères ou de bits considéré comme un tout par l'ordinateur.
MULTI-PROGRAMMATION : Technique d'utilisation d'un ordinateur par laquelle l'exécution de plusieurs programmes peut être menée, soit simultanément, soit en alternance.

O

OBJET : Voir : langage.
OCTAL : Système de numération à base 8.
OCTET : Groupe de 8 bits traité comme un tout par l'ordinateur.
OFF LINE - ON LINE : (hors ligne, en ligne) Mode de connexion de certains périphériques pouvant travailler, soit de manière autonome (off line), soit directement avec l'ordinateur (on line).
OPERANDE : Quantité entrant dans une opération arithmétique ou logique.
OPERATEUR : Symbole représentant un traitement mathématique à effectuer sur les facteurs (opérandes) qu'il associe. On distingue les opérateurs arithmétiques (+, −, ...) ou logiques (et, ou, ...).
OPTIMISATION : Travail ou attitude consistant à atteindre un résultat plus rapidement et plus sûrement.
ORDINATEUR : Calculateur à programme enregistré pour le traitement de l'information. Un ordinateur comprend : une unité centrale de traitement, des unités d'entrée et de sortie de données, et, éventuellement, des dispositifs de mémoire auxiliaire et des éléments terminaux.
ORGANIGRAMME : Représentation graphique d'un algorithme, montrant l'enchaînement des opérations à l'aide de symboles conventionnels.

P

PARITE : Voir : bit de parité.
PERFORATEUR (de cartes, de ruban) : Dispositif périphérique permettant de préparer ou d'éditer des données sur support papier ou carton au moyen de perforations.
PERIPHERIQUE : Elément connecté à l'ordinateur, commandé par lui, permettant d'introduire ou de recueillir des données. Exemples : périphériques mémoires (bandes, disques) ; périphériques à visualisation (écran cathodique,...) ; périphériques d'introduction de données (clavier,...) ; périphériques d'édition (imprimantes,...).
PROCESSEUR : Terme souvent utilisé pour désigner l'unité centrale d'un ordinateur ou un programme de traduction.
PROGRAMMATION : Action consistant à écrire un programme (phases d'analyse, d'écriture et de mise au point).
PROGRAMME : Ensemble des opérations, exprimées sous forme d'instructions, rédigées dans un langage dit de programmation, décrivant le ou les traitements à effectuer, et assimilable, directement ou indirectement par l'ordinateur.
PROTECTION : Dispositif mis en œuvre pour que des zones mémoires soient protégées vis-à-vis des programmes non autorisés ou vis-à-vis d'erreurs de manipulation. Les dispositifs peuvent être de nature logicielle ou matérielle, selon les supports mémoires.

R

REGISTRE : Zone mémoire particulière faisant partie des organes de calcul ou de commande. Les registres sont affectés à des opérations spécifiques, selon leurs types (registres adresses, registres arithmétiques...).

S

SEQUENTIEL : Voir Accès.
SOFTWARE : Voir Logiciel.
SOURCE : Voir langage.
SUPERVISEUR : Programme contrôlant l'exécution d'autres programmes. Voir : Moniteur.
SYNCHRONE : Mode de fonctionnement d'un calculateur ou d'un sous-ensemble (périphérique, terminal) par lequel les opérations s'exécutent en un certain nombre de cycles complets.
SYSTEME : Machine, ordinateur, par extension l'ensemble des composants, périphériques, terminaux connectés à un ordinateur.
SYSTEME D'EXPLOITATION : Ensemble des programmes permettant d'utiliser un ordinateur.

T

TELEINFORMATIQUE : Technique d'utilisation d'un ordinateur à distance. Les postes de travail dits éléments terminaux sont reliés à l'ordinateur central par lignes téléphoniques publiques (réseau commuté) ou privées (lignes louées).
TERMINAL : Poste terminal. Appareil relié à distance à un ordinateur. Exemples : télé-imprimeur, écran cathodique...
TRADUCTEUR : Programme de transcription d'une information d'un langage dans un autre.

U

UNITE : Dispositif d'entrée ou de sortie de données. Exemples : dérouleur de bandes ; lecteur/enregistreur de disques ; écran/clavier...
UNITE CENTRALE : Cœur de l'ordinateur, l'unité centrale réalise les opérations (arithmétiques ou logiques) qui lui sont fournies par programme.

TABLE DES MATIÈRES

Liste des chapitres 5
Note de l'éditeur 7

Chapitre I - Le phénomène informatique 9
1. Les années 50 : naissance de l'ordinateur 9
2. L'arrivée des semi-conducteurs : diodes et transistors 10
3. La troisième génération 10
4. Apparition des micro-ordinateurs 11
5. Qu'est-ce que la micro-informatique ? 12
6. L'informatique personnelle 13

Chapitre II - Informatique et ordinateur 17
1. L'information et le traitement de l'information 17
2. Architecture d'un ordinateur 21
   2.1. La mémoire 21
   2.2. L'unité de traitement 22
   2.3. Les entrées-sorties 23
3. L'ordinateur et les échelles de temps 25

Chapitre III - Les systèmes de codage 27
1. Notion d'information digitale 27
2. Base et système de numérotation 28
   2.1. Le système décimal 28
   2.2. Système à base B 29
   2.3. Formule de définition d'un nombre 30
   2.4. Différents systèmes de numérotation 30
   2.5. Conversion de base à base 32
   2.6. Les nombres fractionnaires 34
3. Arithmétique binaire 36
   3.1. Table d'addition 36
   3.2. Table de multiplication 36
   3.3. Soustraction binaire 37
   3.4. Division binaire 38
4. Numération hexadécimale 39

5. Caractère-mot représentation des nombres dans un ordinateur..... 40
5.1. Notion de caractère..... 41
5.2. Notion de mot..... 41
6. Ecriture en virgule fixe, virgule flottante..... 42
6.1. Notation scientifique..... 43
6.2. Introduction du signe..... 43
7. Codification des informations..... 44
7.1. Le décimal codé binaire ou DCB..... 44
7.2. L'ASCII (ou American Standard Code for Information Interchange..... 46

Chapitre IV - Les opérateurs logiques..... **51**
1. Fonction « ET » (AND)..... 51
2. Fonction « OU » (OR)..... 52
3. Fonction « NON » (NOT)..... 53
4. Fonction « NON ET » (NAND)..... 53
5. Fonction « NON OU » (NOR)..... 54

A l'intérieur du micro-ordinateur..... 55

Chapitre V - Description d'un micro-ordinateur..... **59**
1. La mémoire centrale..... 59
1.1. Notion de mémoire..... 59
1.2. Organisation d'une mémoire..... 60
1.3. Caractéristiques d'une mémoire..... 60
1.4. Classement des mémoires..... 61
1.5. Un peu d'histoire : la mémoire à tores..... 63
1.6. Mémoires à circuits intégrés..... 65
1.7. Technologie des mémoires à circuits intégrés..... 65
2. Fonctionnement de l'unité centrale de traitement..... 69
2.1. Fonctions..... 69
2.2. Le microprocesseur..... 69
2.3. Les instructions..... 70
2.4. L'exécution d'une instruction..... 70
2.5. Enchaînement des instructions : le compteur ordinal..... 71
2.6. Déroulement d'un programme..... 72
2.7. Eléments mis en œuvre..... 72
2.8. Interruption..... 73
3. Communications entre CPU, mémoires et périphériques..... 74

Autour du micro-ordinateur..... 77

Chapitre VI - Les périphériques **81**
1. Périphériques à visualisation 81
1.1. Communication « homme-machine » 82
1.2. La visualisation d'informations 83
1.3. Le tube à rayons cathodiques 84
1.4. Principe des écrans cathodiques 84
1.5. Constitution d'un écran 85
1.6. Affichage des caractères 85
1.7. Types de terminaux à écran cathodique 86
1.8. Autres technologies 86
2. Le clavier 87
2.1. Composition d'un clavier alphanumérique 87
2.2. Touches fonctions 88
2.3. Touches numériques 88
2.4. Fonctions complémentaires : la gestion du curseur 88
3. Périphériques électro-magnétiques 89
3.1. Périphériques et support 89
3.2. Constitution du support : types différents 90
3.3. Bande magnétique 91
3.4. Cassette magnétique 91
3.5. Disques 92
3.6. Organisation des informations sur les disques 94
3.7. Disque souple 96
3.8. La disquette 98
3.9. Unité à disques souples et caractéristiques 99
4. Précautions à prendre dans la manipulation des supports magnétiques 100
5. Sortie papier 101
5.1. La fonction impression 101
5.2. Les imprimantes 101
5.3. Différents types d'imprimantes 102
5.4. Imprimantes à aiguilles 103
5.5. Jeux et générateurs de caractères 106
6. Constitution standard d'une imprimante 107
6.1. Le bloc d'impression 107
6.2. Le service papier 107
6.3. L'électronique de commande 107
7. Autres types de périphériques 108
7.1. Le badge magnétique 108
7.2. Le photostyle, ou crayon lumineux 108
7.3. Le lecteur de codes à barres 108
7.4. L'entrée vocale 108
7.5. Sortie vocale 109
7.6. Sortie graphique 109
7.7. Tables traçantes ou traceurs 109

Chapitre VII - Analyse d'un système : le LX 500 ..... **111**
1. Constitution 111
1.1. Description générale 111
1.2. Caractéristiques physiques et conditions d'installations 112
1.3. L'unité centrale à Z 80 112
1.4. La mémoire principale 113
1.5. La mémoire auxiliaire 114
1.6. Alimentation 115
1.7. Autres caractéristiques 115
1.8. Architecture du LX 500 115
2. Configuration de travail du LX 500 117
2.1. Description de l'écran/clavier 118
2.2. La station d'impression 121
3. Utilisation 121
3.1. Mise en marche du LX 500 121
3.2. Fonctionnement sous systèmes B DOS 122
3.3. Fonctionnement sous Basic 123

Chapitre VIII - Dialogue et programmation **129**
1. L'utilisation active 129
1.1. Comment faire exécuter ? 133
1.2. Numérotation de 10 en 10 133
1.3. L'INPUT 135
1.4. La boucle 137
1.5. Du nom du client 138
2. D'une représentation graphique du programme 144
3. Le test 145
3.1. L'instruction IF 148
3.2. Test de fin de boucle 149
4. La numérotation des instructions 151
5. Notion de sommation 152

Chapitre IX - Le système d'exploitation **155**
1. Les trois programmes 155
1.1. Principales commandes du B DOS 155
1.2. Principales commandes du BASIC 157
1.3. Commande du programme utilisateur 157
1.4. Commandes extérieures 157
2. Sauvegarde d'un programme et utilisation du B DOS 157
3. Utilisation du BASIC 160

Chapitre X - Les programmes et leur exécution **165**
1. Modes d'utilisation d'un micro-ordinateur 165
1.1. La monoprogrammation 165
1.2. La multi-programmation 169
1.3. Le temps partagé 170
1.4. Le multi-traitement 170
1.5. Le temps réel 172
2. Système d'exploitation 173
2.1. Programmes privilégiés 174
2.2. Programmes non privilégiés 174
2.3. Notion de traducteurs de langage 174
2.4. Système de programmation 176

Chapitre XI - Programmons en BASIC **179**
1. Calculette algébrique 180
2. Quelles données numériques sont acceptées par Basic 182
3. Et quelles données alphanumériques ? 183
4. Utilisation du « mode bureau » sur un programme 184
5. Temps perdu ne se rattrappe guère 185
5.1. Instruction INPUT 186
5.2. Affectation 186
5.3. Instruction PRINT 187
6. Bonjour monsieur, bonjour madame 188
6.1. Chaîne de caractère 188
6.2. Sous-chaîne 190
6.3. L'instruction IF 190
7. Parquez vos moutons 192
7.1. L'itération : la boucle de 1 à N 193
7.2. Recherche d'un maximum 194
7.3. Généralisation de la boucle FOR BASIC 195
8. Température de l'eau du robinet 196
8.1. Notion de tableau 197
8.2. Calcul de s = t(a) + t(a+1)+... +(b−1) 199
9. Pluriel des noms en « OU » 200
9.1. Travail sur sous-chaînes 200
9.2. Retrouver un mot dans une chaîne 202
9.3. Pluriel 206
10. Calcul d'engrenage 209
10.1. Recherche d'un couple R1, R3 212
10.2. Vérification des contraintes 213
Solution des exercices 214

Chapitre XII - Quelques programmes pratiques en Basic **221**
1. Extraction de racines carrées et cubiques 221
1.1. L'algorithme de base pour racines carrées 222
1.2. Avec l'instruction ABS 225
1.3. Extraction de la racine cubique 226
2. Chaînes de caractères les 7 jours de la semaine 229
3. Algorithme de tri numérique 231
4. Le tracé de sinus 234
4.1. Calcul des valeurs de sinus x 234
4.2. Tracé avec l'ordre TAB 236
4.3. Si votre Basic ne dispose pas de l'ordre TAB 238
5. Les carrés magiques 239

Chapitre XIII - Du Basic au LSE **247**
1. Structure d'un programme 248
1.1. Un programme 248
1.2. Une instruction 249
2. Représentation des données 250
2.1. Les constantes 251
2.2. Les variables 253
2.3. Les tableaux 254
3. Les opérateurs de traitement 256
3.1. Opérateurs numériques 256
3.2. Opérateurs sur chaîne 258
3.3. Les expressions 259
4. Structure des instructions 260
4.1. Les déclarations 260
4.2. L'affectation 261
4.3. Les instructions d'entrée-sortie 261
4.4. Les branchements 262
5. Les procédures et sous-programmes 265
5.1. En Basic 266
En LSE 267
6. Initialisation de l'ordinateur et manipulation du terminal 268

Chapitre XIV - Quelques programmes pratiques en LSE **271**
1. Bon courage 271
2. Calcul de l'aire de triangles 272
3. Rangement de nombres par ordre croissant 274
4. Ce nombre est-il premier ? 278
5. Le jeu de Marienbad 281

Chapitre XV - Maintenance et fiabilité **287**
1. Introduction à la notion de maintenance 287
1.1. La garantie 288
1.2. La maintenance en informatique 288
1.3. Installation 289
1.4. Matériel et logiciel 290
2. Le contrat de maintenance 292
2.1. La location 292
2.2. La maintenance au forfait 292
2.3. La maintenance au coup par coup 293
2.4. Le contrat d'entretien 293
2.5. Paramètre à prendre en compte 293
3. Maintenance préventive-curative 294
3.1. Maintenance curative 294
3.2. Maintenance préventive 294
4. Maintenance et fiabilité 295
4.1. Fiabilité d'un élément 295
4.2. Fiabilité du système 296
4.3. MTBF 296
4.4. MTTR 298

Annexes **303**
1. Le langage Basic au complet 303
1.1. Les instructions 303
1.2. Instructions de branchement 303
1.3. Les fichiers 304
1.4. Les fonctions 304
2. L'interprétateur, l'éditeur, et le B DOS 304
3. Petit vocabulaire du traitement de l'information 305

Imprimé en France par Berger-Levrault, Nancy.
Dépôt légal n° 6160-2-1983
Impression n° 779965 16/5129/8
Collection n° 42. Édition 03.

comprendre la

MICRO-INFORMATIQUE

PRATIQUE DES MICRO-ORDINATEURS

35 ans après la construction du premier ordinateur, l'informatique déjà présente dans quelques lycées expérimentaux, ainsi que dans bon nombre de lycées techniques et de lycées d'enseignement professionnel, entre en force dans les établissements secondaires où, déjà, un grand nombre de micro-ordinateurs ont été mis en place.

Il ne s'agit pour l'instant, ni d'une nouvelle matière à enseigner, ni d'une nouvelle manière d'enseigner, mais de la prise en compte dans l'éducation d'un phénomène majeur qui est en train de bouleverser notre société.

Ce livre se veut ouvrage de référence et ouvrage d'initiation : les notions qui sont développées ici, permettent au lecteur, en fonction de ses préoccupations, soit d'acquérir une culture de base sur l'informatique ou la connaissance des principes et des termes techniques, soit de disposer des éléments d'un savoir-faire suffisant pour l'utilisation courante d'un micro-ordinateur dans l'enseignement.

LORRAINS
Réseaux téléinformatiques
HACHETTE

A. OSBORNE
Initiation aux micro-ordinateurs
S.E.C.F. ÉDITIONS RADIO

P. MORVAN
L'ordinateur et l'informatique
S.E.C.F. ÉDITIONS RADIO

H. LILEN
L'ordinateur personnel
S.E.C.F. ÉDITIONS RADIO

9 782010 078323

16 5129 8

84,00 FF TTC